SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Luka Kurelja

Zagreb, 2018.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Mentori:

Prof. dr. sc. Bojan Jerbić, dipl. ing.

Student:

Luka Kurelja

Zagreb, 2018.

Izjavljujem da sam ovaj rad izradio samostalno koristeći znanja stečena tijekom studija i navedenu literaturu.

Zahvaljujem se mentoru prof. dr. sc. Bojanu Jerbiću na pomoći, savjetima i sugestijama tijekom izrade diplomskog rada te Filipu Šuligoju mag. ing. na strpljenju, smjernicama i uloženom vremenu. Također, veliko hvala firmi KOPACK d.o.o. na posuđenoj opremi koju sam koristio pri izradi rada kao i g. Ervinu Palanoviću na pomoći oko komunikacijskih protokola opreme.

Posebno se zahvaljujem svojim roditeljima Deanu i Katiji Kurelja te bratu i sestri na pruženoj podršci, strpljenju, razumijevanju i pomoći tijekom dosadašnjeg života, a posebno tijekom studijskog obrazovanja.

Luka Kurelja

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE
Središnje povjerenstvo za završne i diplomske ispite
Povjerenstvo za diplomske radove studija strojarstva za smjerove:
proizvodno inženjerstvo, računalno inženjerstvo, industrijsko inženjerstvo i menadžment,
inženjerstvo materijala te mehatronika i robotika

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum:	Prilog:
Klasa:	
Ur. broj:	

DIPLOMSKI ZADATAK

Student: **LUKA KURELJA** Mat. br.: 0035177457

Naslov rada na hrvatskom jeziku: **Sinkrono upravljanje automatskom linijom za pakiranje**

Naslov rada na engleskom jeziku: **Synchronous control of automatic packaging line**

Opis zadatka:

Automatsko pakiranje prehrambenih proizvoda provodi se na pokretnim stazama koje karakterizira sinkrono upravljanje pogonskim motorima (tzv. "Flow Pack" sustavi). Za potrebe ovog diplomskog rada potrebno je projektirati, izraditi i programirati laboratorijsku maketu koja simulira rad "Flow Pack" sustava. Maketa treba integrirati tri motora u sinkronom načinu rada. Zadatak prvog motora je simulacija kretanja trake po kojoj predmeti putuju do mjesta pakiranja. Drugi motor pokreće valjke koji izvlače iz role gornju i donju stranu pakirne folije i zavaruju rubove oblikujući prostor u koji se predmeti s trake umeću. Treći motor pokreće nož koji reže foliju na točnu mjeru okomito na smjer kretanja iste. Izrađena maketa treba omogućiti korištenje: EMMS-AS motora, CMMP-AS-M3 motor kontrolera i dva PLC-a (CECX-X-M1 I CDPX). Potrebno je programirati i ostvariti sinkronizaciju sva tri motora praćenjem pozicije vodećeg motora preko čije se vrijednosti zadaju pozicije i brzine druga dva motora. Programsko sučelje mora operateru makete omogućiti odabir predefiniranih dimenzija predmeta rada i brzinu kretanja trake. U radu je potrebno navesti korištenu literaturu i eventualno dobivenu pomoć.

Zadatak zadan:
03. svibnja 2018.

Rok predaje rada:
05. srpnja 2018.

Predviđeni datum obrane:
11. srpnja 2018.
12. srpnja 2018.
13. srpnja 2018.

Predsjednica Povjerenstva:
prof. dr. sc. Biserka Runje

Zadatak zadao:
prof. dr. sc. Bojan Jerbić

SADRŽAJ

POPIS SLIKA

POPIS TABLICA

POPIS TEHNIČKE DOKUMENTACIJE

1 Desna ploča korita
2 Lijeva ploča korita
3 Bazna ploča korita
4 Namotna Osovina Prvi Dio
5 Namotna Osovina Drugi Dio
6 Natezna Osovina
7 Osovina Noža
8 Osovina Olovke
9 Osovina Za Papir Prvi Dio
10 Osovina Za Papir Drugi Dio
11 Drzač za olovku
12 Brid noža
13 Maketa

POPIS OZNAKA

Oznaka	Jedinica	Opis
DC	mm	Duljina proizvoda
R	mm	Duljina razmaka između proizvoda
C	mm	Zbroj duljine proizvoda i razmaka
$v_{o,z}$	mm/s	Obodna brzina osovine M3 motora
$v_{o,r}$	mm/s	Referentna obodna brzina za koju osovina napravi jedan krug u minuti
$v_{co,r}$	ink/s	Referentna brzina u CoDeSys-u za koju osovina napravi jedan krug u minuti
$v_{co,z}$	ink/s	Zadana brzina u CoDeSys-u
t_p	s	Vrijeme potrebno da M1 napravi puni krug
ω_p	s^{-1}	Kutna brzina
ω_r	s^{-1}	Referentna kutna brzina za koju osovina napravi jedan krug u minuti
α	°	Kut rotacije
φ	-	Kut rotacije
N_d	mm	Pomak do aktivacije M1 motora
L_k	mm	Udaljenost od točke crtanja do točke rezanja
d	mm	Pomak
N_{di}	ink	Pomaka u CoDeSys-ovim jedinicama

SAŽETAK

Diplomski rad istražuje mogućnosti sinkronog upravljanja sa servomotorima pomoću PLC-a. Sinkronizacija i ispitivanja su rađena na Flow pack stroju, točnije na eksperimentalnom postavu koji simulira njegov rad.

Flow pack je stroj za pakiranje proizvoda, koji na principu protočnog umatanja i pakira proizvode u polipropilensku foliju. Proizvodi koji se ovako pakiraju nemaju uvjet oblika, ali imaju uvjete na dimenzije.

Napravljeni eksperimentalni postav simulira rad pravog stroja i ovdje su proizvodi predstavljeni jednodimenzionalno s nacrtanom linijom na pokretnoj papirnoj traci. U radu su korištena tri motora, dok pravi Flow pack može koristiti i više. Sinkronizacija je napravljena tako da jedan od motora 'vodi' druga dva. Preciznije rečeno, na temelju položaja i brzine vodećeg motora, određuju se pozicije i brzine druga dva.

U prvom poglavlju je kratak uvod u automatizaciju i temu rada. Drugo poglavlje daje pregled korištene opreme za izradu eksperimentalnog postava. U trećem su obrađeni tipovi komunikacije među komponentama sustava, dok je u četvrtom navedena sva korištena programska podrška i opisan je rad programa. Peto poglavlje opisuje konceptualno rješenje problema zajedno s matematičkom podlogom koja omogućuje rad istog, uz to prikazani su i komentirani dobiveni rezultati. Šesto poglavlje daje pregled grafičkog upravljačkog sučelja s kojim operater zadaje predefinirane parametre upravljanja, pokreće i zaustavlja sustav.

Ključne riječi: automatizacija, PLC, sinkronizacija, upravljanje, servomotori, FESTO

SUMMARY

This master thesis explores possibilities of synchronous control of servomotors using PLC. For that purpose experimental setup/model was made, which simulates Flow pack machine.

Flow pack is a machine which uses flow wrapping to package products. Flow wrapping is process in which product of any shape is wrapped in polypropylene film.

Made model of flow pack uses paper strip on which line is drawn that represents the product. Three servomotors used in this setup, of which one is a lead motor. Based on its positioning and speed, positions and speeds of other two are calculated and set.

First chapter gives a brief introduction into automation and theme of this thesis. Second chapter describes equipment used in making of the model. Communication types are described in third chapter, while fourth one gives an overview of used software and elaborates how the program functions. Next is the fifth chapter, in which are given mathematical equations and conceptual solution for system control along with test results. Sixth chapter is an overview of graphical user interface. Through which operator can set control parameters, initiate or stop the system.

Key words: automation, PLC, synchronization, control, servomotors, FESTO

1. UVOD

U svim granama industrije glavni cilj uvijek je bio proizvesti robu na brži, efikasniji i pametniji način. To je dovelo do prvih dviju industrijskih revolucija i trenutno pokreće treću.

U današnje vrijeme svi proizvodni procesi pokušavaju se automatizirati u većoj ili manjoj mjeri. Industrijska automatizacija je upotreba upravljačkih sustava kao što su: računala, PLC-i i mikro kontroleri kako bi se upravljalo i regulirao raznim procesima i strojevima. Prednost automatiziranih sustava su: ponovljivost, brzina, ujednačenost kvalitete proizvoda itd., dok su nedostatci: visoka početna cijena sustava, nepredvidive i visoke cijene razvoja, smanjenje radnih mjesta itd. Veći zahtjevi u automatizaciji su fleksibilnost i konvertibilnost proizvodnih procesa kojima proizvođači žele, uz male preinake u sustavu, na istoj proizvodnoj liniji proizvoditi dva različita proizvoda. Tome je jako pridonio razvoj digitalne elektronike i Fieldbus. Fieldbus je industrijska računalna mreža koja spaja sve upravljačke jedinice preko jednog kabela i standardizirana je prema ICE 61158 standardu.

Najčešće korišten upravljački sustav industrijskih procesa je PLC (Programibilno logički kontroler). PLC je specijalizirano računalo za upravljanje industrijskih procesa i nasljednik je relejne logike. Njegova glavna uloga je čitanje i obrada ulaznih podataka te postavljanje vrijednosti izlaza. Ulazni podatci se najčešće primaju s različitih senzora, dok se izlazni podatci šalju na motore, cilindre ili se prosljeđuju drugom PLC-u. Razlikuju se dvije glavne skupine PLC-a: integrirani i modularni. Integrirani PLC je sastavljen od nekoliko različitih modula koji su svi stavljeni u jedno kućište te je time ulaze i izlaze odredio proizvođač. Modularni PLC-ovi imaju jedan centralni modul (CM) u kojemu se nalazi napajanje i procesor te se na njega spajaju različiti ulazno/izlazni moduli, a oni mogu biti digitalni, analogni ili specijalizirani, no to ovisi o vrsti korištenih senzora u sustavu. Broj spojenih modula je ograničen napajanjem CM-a, ali zato postoje moduli koji dovode dodatno napajanje ako je proširenje iznad maksimalnog broja modula koje CM može podržati.

Ulazni signali na PLC dolaze sa senzora, a oni mogu biti: temperaturni, svjetlosni, ultrazvučni, infracrveni, senzori brzine, položaja, akceleracije itd. Izlazni podatci šalju se na: električne, servo i hidraulične motore i/ili na pneumatske i/ili hidrauličke cilindre.

Automatizirani industrijski procesi smanjuju broj potrebnih radnika, ali povećavaju prag količine znanja i vještina potrebnih kod preostalih radnika. Zbog smanjenja ljudskog faktora u

proizvodnom procesu, javlja se potreba za umrežavanjem i međusobnom komunikacijom među različitim strojevima u industrijskim pogonima. Komunikacija među strojevima omogućava sinkronizaciju njihovog rada i glavni je faktor glatkog rada cijelog proizvodnog postava. Svaki stroj proizvodnog procesa je zaseban sustav, unutar kojeg se izvršava sinkronizacija među pokretnim komponentama.

U ovom radu napravljena je sinkronizacija motora na maketi Flow pack stroja. Stroj se koristi za pakiranje proizvoda u polipropilensku foliju. Neki od proizvođača ovakve opreme su CHLB, FANT, PAXIOM Group i drugi. Razlikuju se vertikalni i horizontalni Flow pack strojevi. Naredni tekst opisuje princip rada horizontalnog stroja.

Princip rada je sljedeći: proizvodi od mjesta utovara do mjesta pakiranja putuju ujednačenom brzinom i na konstantnom razmaku. Na mjestu pakiranja ulijeću u tunel od polipropilenske folije i nastavljaju putovati u njoj do mjesta gdje ju nož reže na dužinu proizvoda.

Slika 1. Flow pack SLEEK40

Gornja slika prikazuje jedan je od četiriju Flow pack-ova te serije SLEEK proizvođača PAXIOM Group. SLEEK40 može zapakirati do 100 proizvoda u minuti sa maksimalnom širinom folije od 400 mm. Pogone ga četiri servo motora koji osiguravaju preciznost pozicioniranja, glatko ubrzavanje i usporavanje tijekom rada.

Izrađena maketa razlikuje se u tome što koristi tri servo motora, zacrtava duljinu proizvoda na papirnatu traku i između njih se označavaju mjesta rezanja. Prvi motor povlači papirnu traku, drugi crta proizvod i treći označava rezove. Motor koji ima istu funkciju

na maketi i na pravom stroju je samo onaj koji pokreće nož, druga dva su prilagođena potrebama makete. Sinkronizacija motora jedan je dio ovog zadatka, dok je drugi dio izrada i dizajn grafičkog sučelja za upravljanje. Grafičko sučelje mora biti jednostavno izrađeno zbog lakšeg rukovanja. Sučelje mora omogućavati promjenu brzine kretanja papira i duljine proizvoda.

1.1. Zadatci

Kako bi se napravila sinkronizacija rada motora i izradilo grafičko sučelje potrebno je izvršiti niz zadataka:

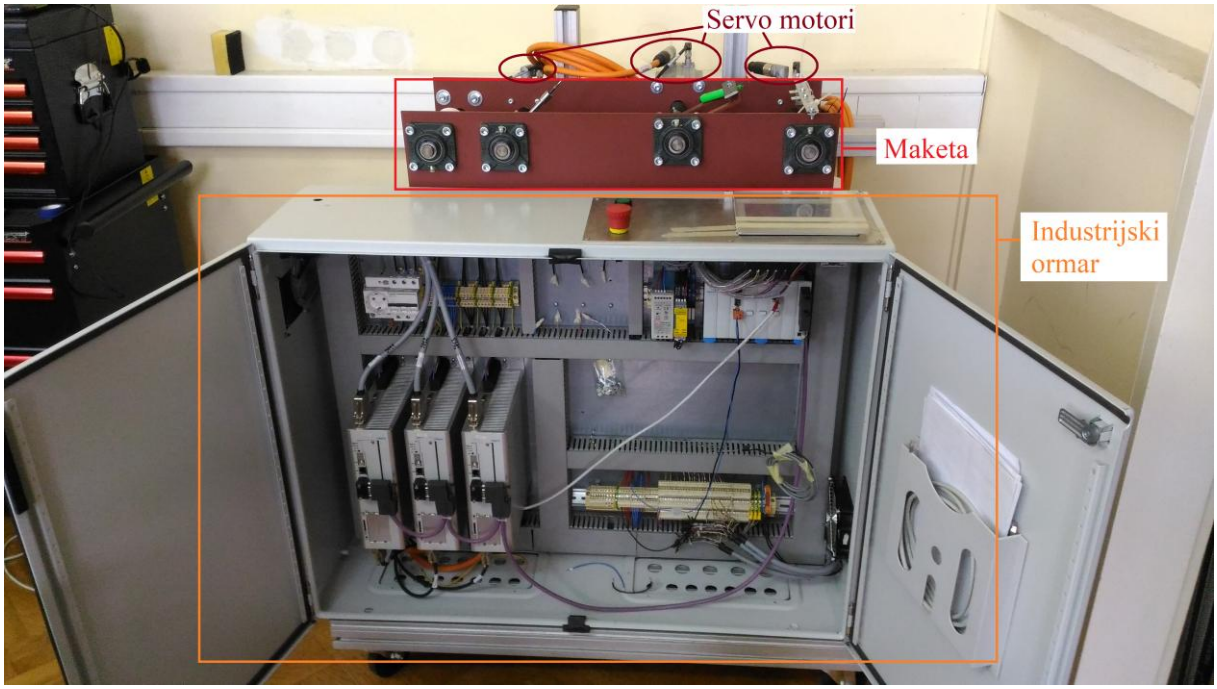
a) dizajnirati i projektirati Flow pack maketu
b) proučiti i spojiti komponente dobivene elektronike
c) parametrizirati motor kontrolere
d) uspostaviti CAN komunikaciju između motor kontrolera i PLC-a
e) razviti algoritam upravljanja za PLC
f) napravit grafičko sučelje za operatorsko rukovanje.

1.2. Ciljevi

Cilj rada je postaviti osnovne principe sinkronizacije motora, na čijim se temeljima može nastaviti daljnji rad i istraživanje.

2. EKSPERIMENTALNI POSTAV

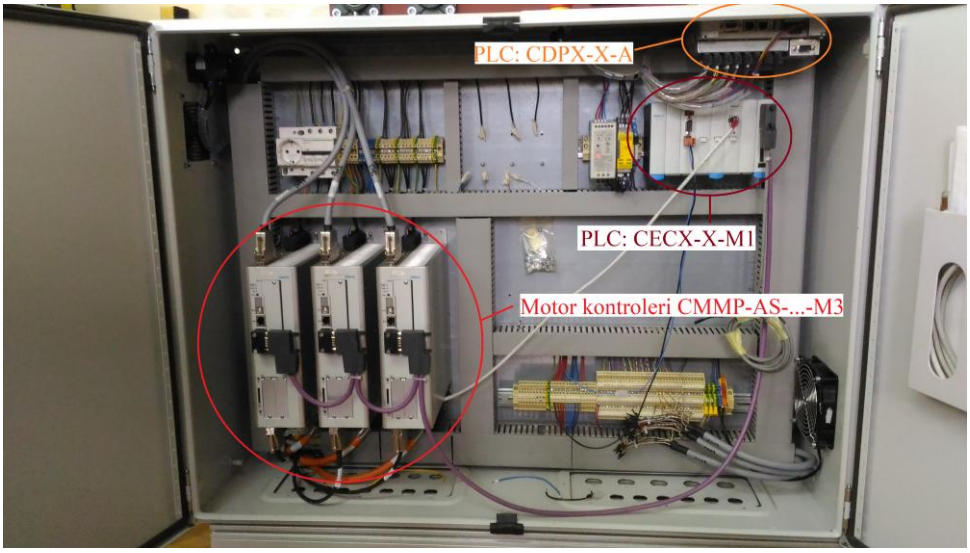
Eksperimentalni postav sastoji se od makete, industrijskog ormara i servomotora. Maketa Flow packa posebno je osmišljena i izrađena za potrebe ovog diplomskog rada. Tehnička dokumentacija ili nacrti makete nalaze se u prilogu. Ostale dijelove postava, motore i ormar, osigurala je firma Kopac d.o.o., a proizvođač opreme je Festo.



Slika 2. Eksperimentalni postav

2.1. Festo oprema

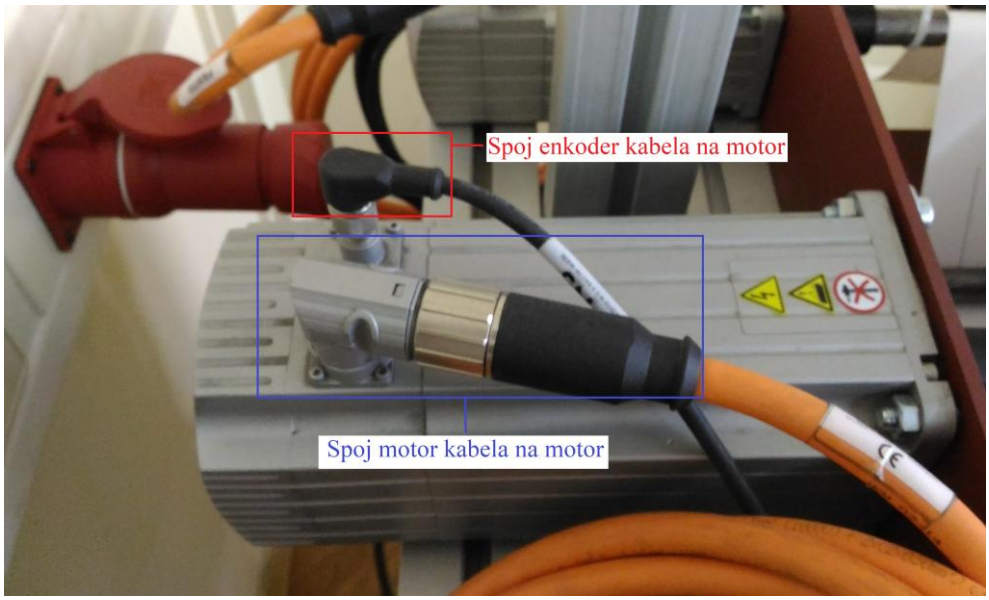
Od Festo opreme u ovom radu korištena su tri motora serije EMMS-AS, tri motor kontrolera serije CMMP-AS i dva PLC-a. Prvi serije CECX i drugi iz serije CDPX. Sva oprema, izuzev motora, spojena je i složena unutar industrijskog ormara.



Slika 3. Elektronika u industrijskom ormaru

2.1.1. Motori

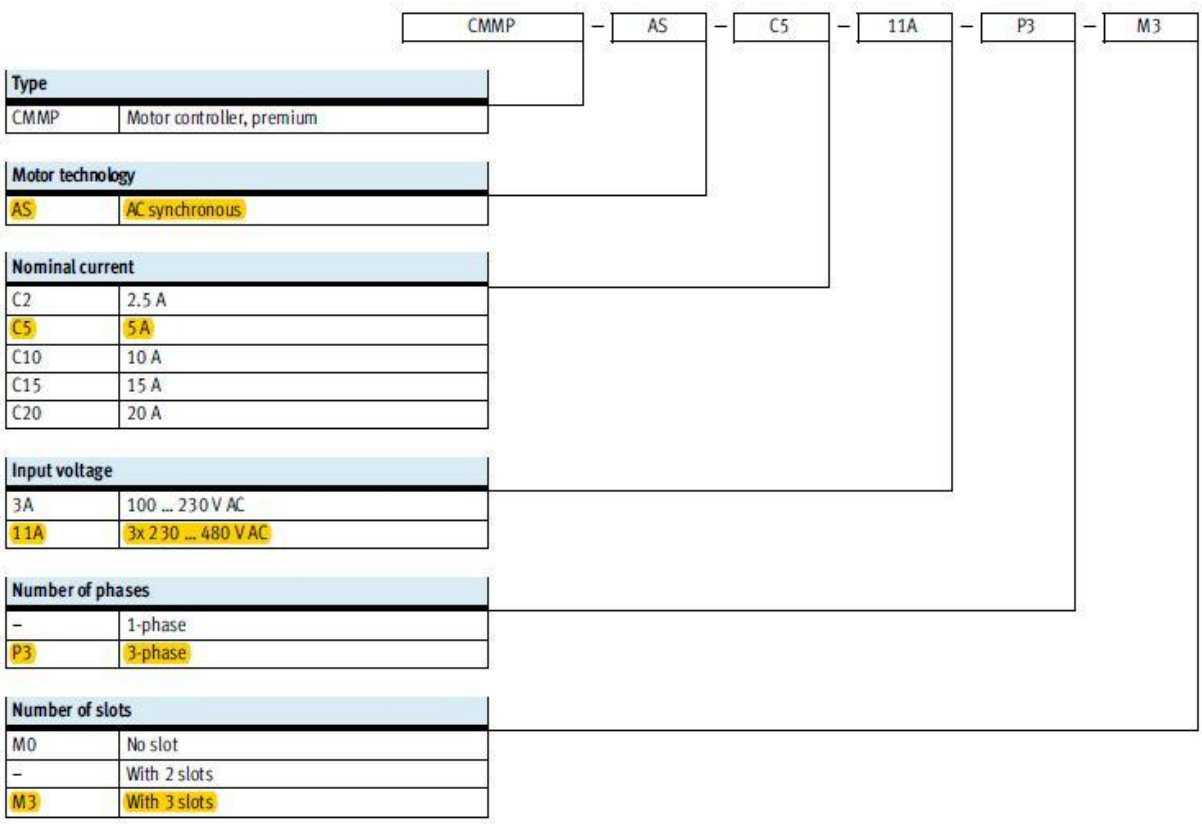
Korišteni su motori serije EMMS-AS, točnije korištena su dva motora oznake EMMS-AS-70-M-HS-RS i jedan oznake EMMS-AS-100-M-HS-RS. Više o motorima se može naći u službenoj dokumentaciji [5]. Motori sa oznakom 70-M pokreću osovine za crtanje proizvoda i pokretanje papira, dok motor 100-M pokreće nož. Svaki motor je spojen na jedan motor kontroler s dva kabela, a to su motor (narančasti) i enkoder (crni) kabel.



Slika 4. Spoj motor i enkoder kabela na motor

2.1.2. Motor kontroleri

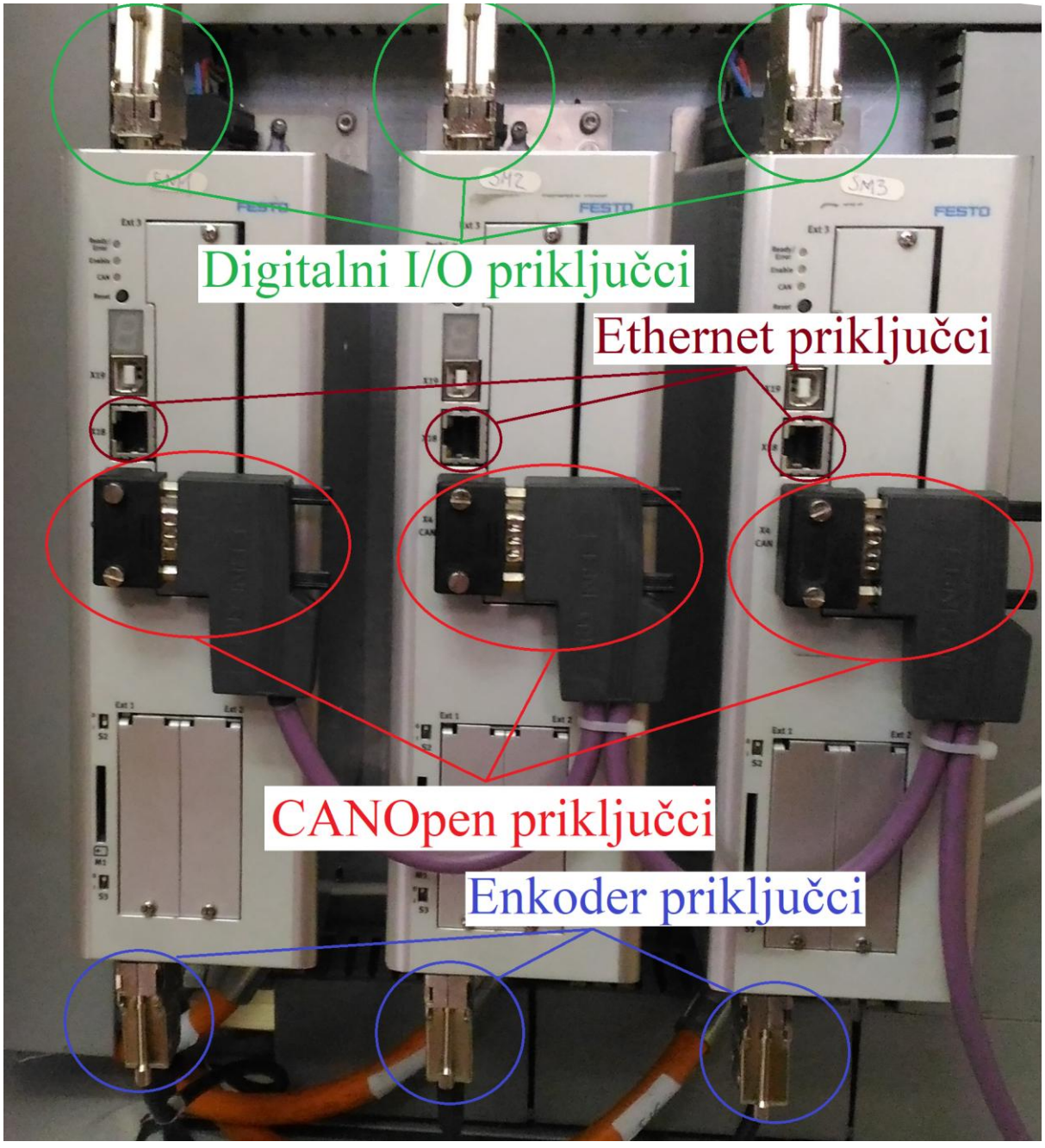
Motor kontroleri služe za izravno slanje upravljačkih signala na motor i primanje podataka s enkodera. U ovom slučaju su korištena tri motor kontrolera iz serije CMMP-AS, njihova puna oznaka je CMMP-AS-C5-11A-P3-M3. Naredna slika pokazuje opis gornje oznake kontrolera. Više o motor kontrolerima u [6] i [7].



Slika 5. Opis za oznake motor kontrolera

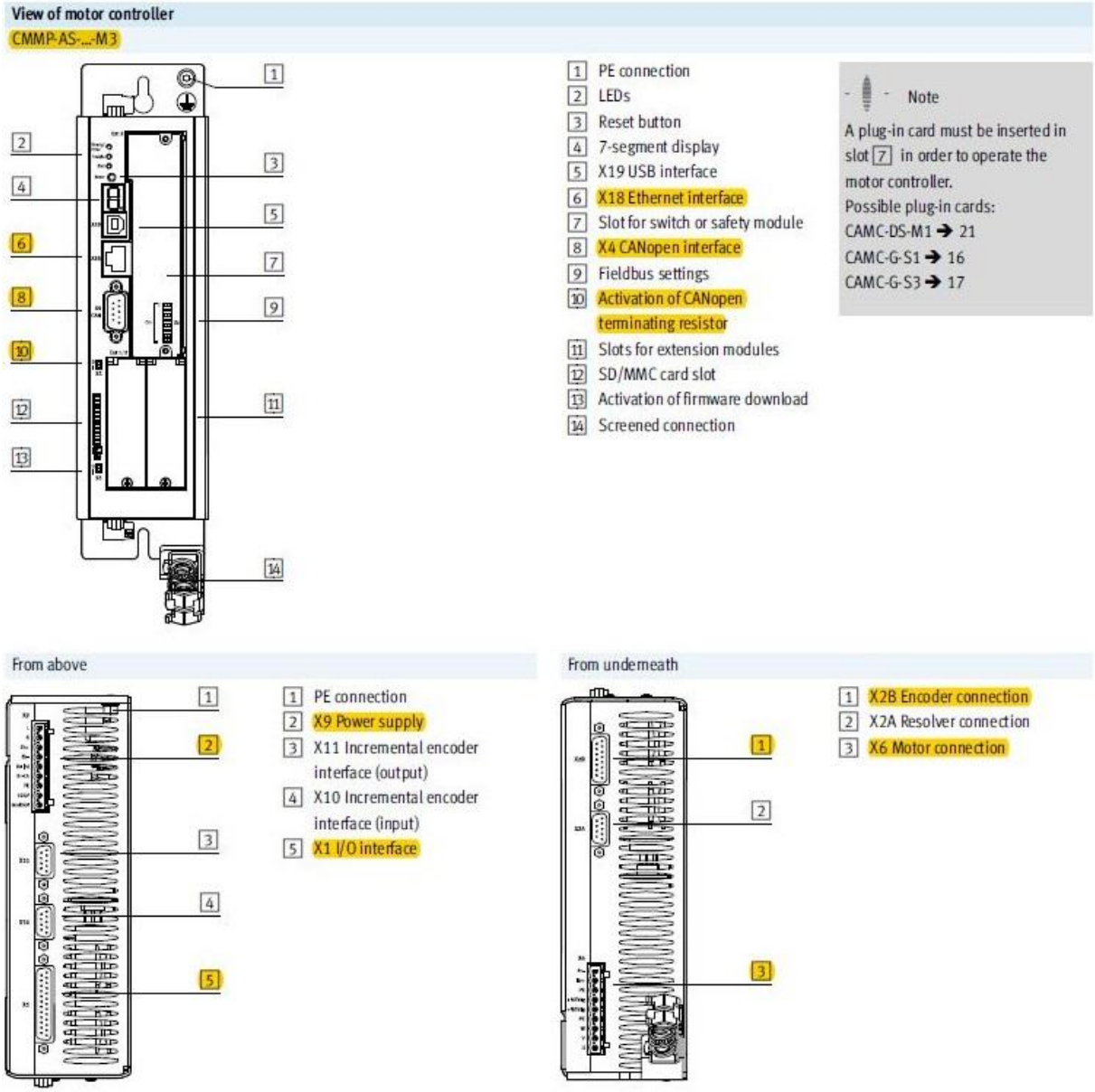
Podešavanje parametara motor kontrolera radi se sa Festo Configuration Tool-om (FCT). Osim parametrizacije, FCT-om se može napraviti Homing motora što ih dovodi u stanje spremno za pokretanje i izravno upravljanje. Ovisno o odabranim parametrima razlikuju se tri načina upravljanja preko FCT-a: upravljanje preko pozicije, brzine i okretnog momenta.

Na slici 6 prikazani su motor kontroleri složeni unutar ormara sa obilježenim spojenim priključcima. Ostatak korištenih priključaka prikazuje slika 7.



Slika 6. Motor kontroleri CMMP-AS...-M3 spojeni u ormaru

Donja slika prikazuje motor kontrolere iz tri pogleda, sa svim njegovim priključcima i indikatorima stanja. U prednjem pogledu žutim bojom označeni su svi priključci na koje se mora obratiti pozornost. Ti priključci su sljedeći: CANopen(X4), Ethernet(X18) priključke i aktivacijsku sklopku otpornika za CAN komunikaciju, a kod pogleda odozgo važno je obratiti pozornost na priključke za napon (X9) i I/O (X1). Zadnji pogled je odozdo i tu su bitni priključci za enkoder (X2B) i za motor kable (X6).

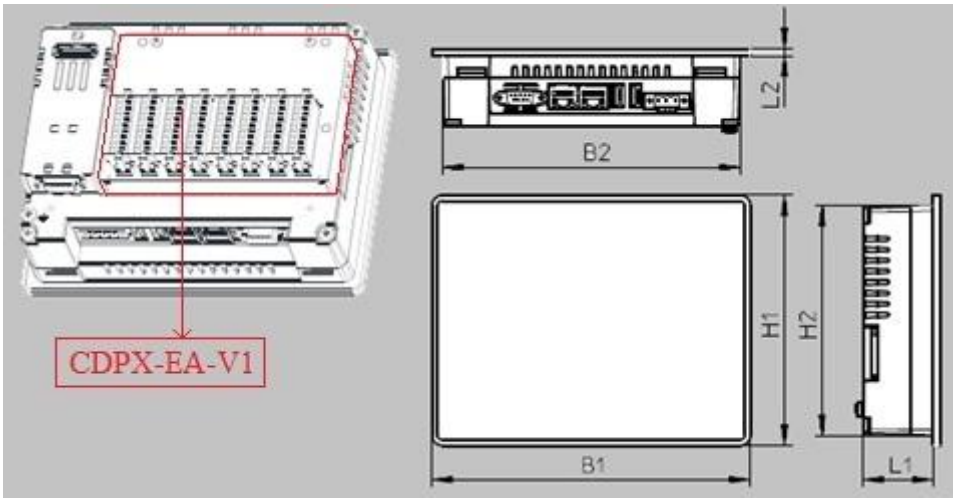


Slika 7. Pogled na motor kontroler iz tri perspektive

Od svih navedenih priključaka treba spomenuti da se Ethernet koristi za parametrizaciju kontrolera, a preko CANopen-a se kontroler povezuje na CAN mrežu sustava. Dok se sa I/O čitaju i zapisuju vrijednosti digitalnih ulaza i izlaza.

2.1.3. *PLC*

U izradi rada korištena su dva PLC-a. Prvi je iz CDPX-X-A, a drugi iz CECX serije. Na CDPX-X-A je spojen modul CDPX-EA-V1 koji dodaje ulazno/izlazne digitalno/analogne priključke.



Slika 8. Pogled na CDPX iz tri perspektive i njegov modularni dodatak CDPX-EA-V1

CDPX-X-A dolazi u četiri verzije: CDPX-X-A-W-4, CDPX-X-A-W-7, CDPX-X-A-S-10 i CDPX-X-A-W-13. Verzije se međusobno razlikuju u dimenzijama označenim na prethodnoj slici i veličini ekrana.

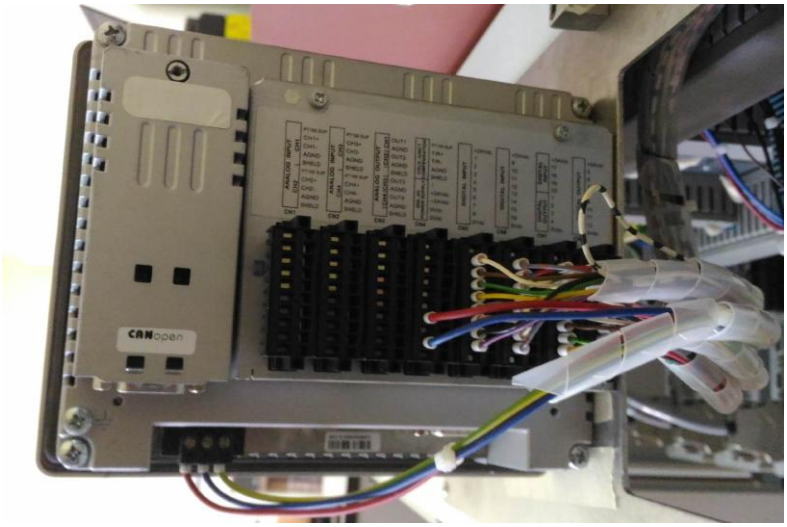
CDPX-X-A – [ ] – [ ]

Function	
CDPX-X-A	Operator unit

Display size, equipment	
S	Standard display, touchscreen
10	10.4", 64 k colours
W	Wide-screen display, touchscreen
4	4.3", 64 k colours
7	7", 64 k colours
13	13.3", 64 k colours

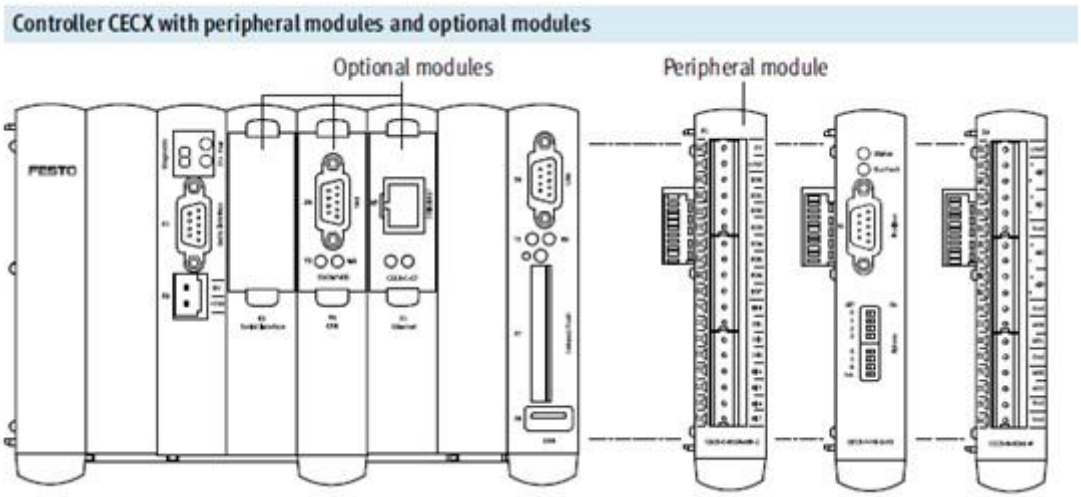
Slika 9. Opis za oznake CDPX-X-A

Slika 9 daje uvid na značenje pojedinih oznaka u nazivu za verzije CDPX-X-A PLC-a. Na sljedećoj slici se vidi realno stanje spojenih digitalnih ulaza/izlaza na modul CDPX-EA-V1, te kako je on spojen na CDPX-X-A.



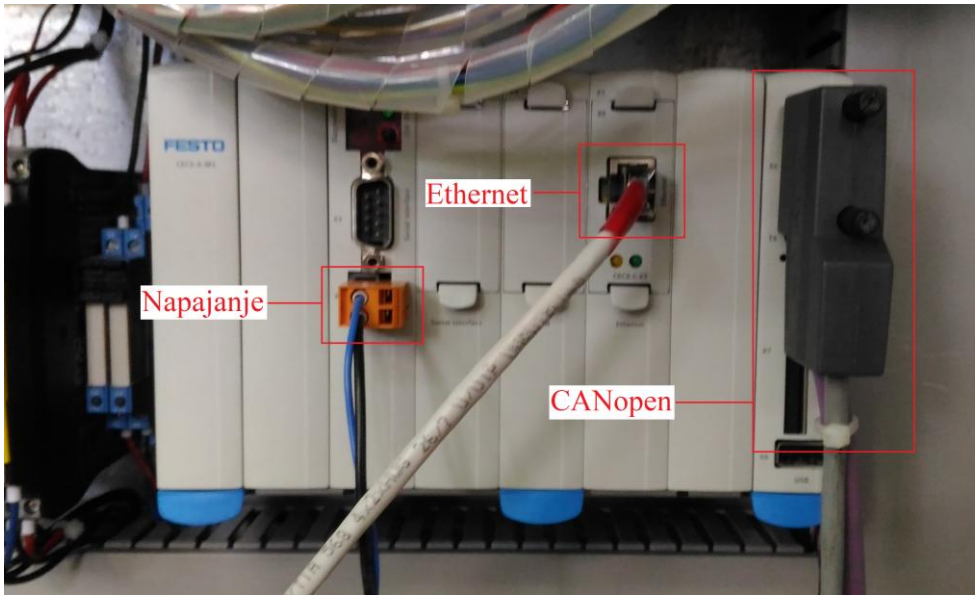
Slika 10. Spoj digitalnih signala na CDPX-EA-V1

CECX je modularni motion PLC, na koji se mogu nadodavati potrebni moduli koji sadrže digitalne ili analogne ulaze i/ili izlaze, ulaze za temperaturne senzore, dodatne priključke za CANopen ili Ethernet ili Profibus ili serijski. Više o ovome kontroleru može se naći pod [1].



Slika 11. Prikaz CECX-X-M1 sa dodatnim i perifernim modulima

Za izradu ovog rada nisu korišteni ni dodatni ni periferni moduli na CECX. On je upravljački PLC koji s motor kontrolerima komunicira samo preko CAN mreže, dok CDPX-X-A upravlja sa svim digitalnim ulazima i izlazima.



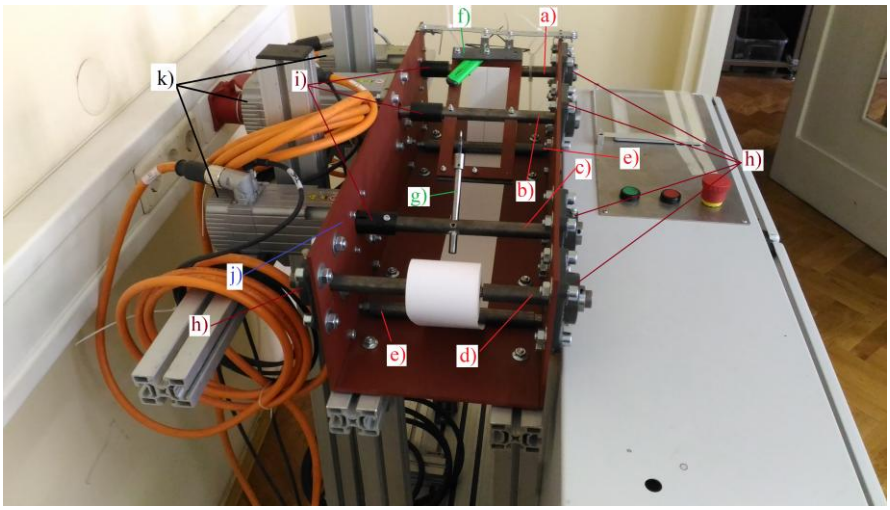
Slika 12. CECX-X-M1 sa svim spojenim priključcima

Oba se PLC-a programiraju preko Ethernet-a, s time da se istim upravljački PLC spaja na grafičko operatorsko sučelje.

2.2. Maketa

Maketa Flow packa napravljena je u obliku korita sa poprečnim osovinama.. Korito makete napravljeno je iz tri čelične ploče. Ploče su zajedno povezane kutnicima, a cijelo korito leži na paketnom stolu. Zbog načina konstrukcije bočne ploče nisu okomite s obzirom na baznu ploču jer to dovodi do problema paralelnosti osovina sa baznom pločom, iz razloga što se neke od osovina sa spojnicama moraju spojiti na motore. Ovaj se problem riješio postavljanjem samopodesivih ležajeva. Razlikuju se tri vrste osovina: pogonske, natezne i jedna pasivna. Pogonske osovine su s jedne strane spojene na servo motore preko spojnica, a s druge su umetnute u ležajeve. Prva pogonska osovina služi za povlačenje papirne trake, druga ima montiran nož i na trećoj se nalazi olovka. Natezne osovine služe kako bi papirnatu traku držale napetom i pridržavale je na dnu korita. Pasivna osovina je na obje strane korita

umetnuta u ležajeve i služi za držanje špule papirne trake. Servo motori su pričvršćeni za maketu sa četiri vijka i leže na dvije poprečne grede.



Slika 13. Maketa

Naredna tablica popis je dijelova označenih na gornjoj slici.

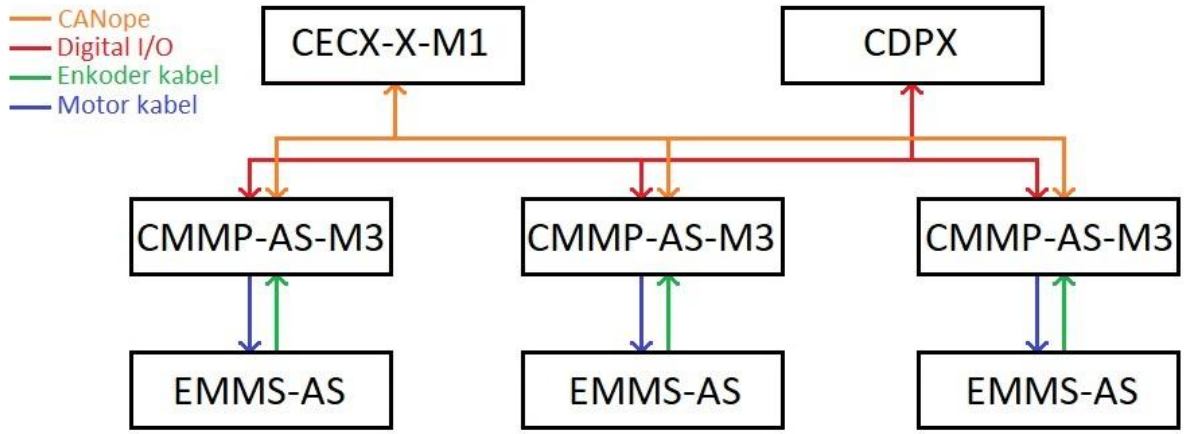
Tablica 1. Dijelovi makete

Oznaka	Dio makete
a)	Prva pogonska osovina
b)	Druga pogonska osovina
c)	Treća pogonska osovina
d)	Pasivna osovina
e)	Natezne osovine
f)	Nož
g)	Olovka
h)	Ležajevi
i)	Sponice motor-osovina
j)	Korito
k)	Motori

3. KOMUNIKACIJA

Postoje dva glavna komunikacijska kanala između PLC-a i motor kontrolera. Prvi je CANopen kanal, a drugi je komunikacija preko I/O priključka. Zbog upotrebe dva PLC-a svaki je na jednom komunikacijskom kanalu; iako su kanali odvojeni, međusobno se nadopunjuju. Napravljeni sustav ne bi mogao funkcionirati samo s jednim od njih. CECX je spojen preko CANopen-a, a CDPX preko I/O-a.

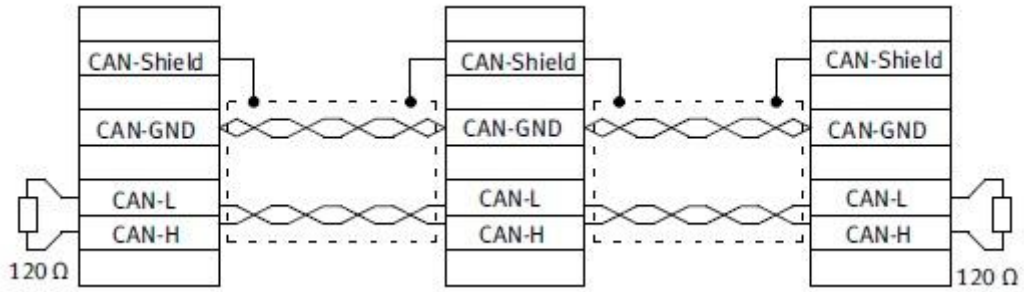
PLC spojen sa digitalnim I/O na kontrolere služi samo za podizanje glavnih bitova DIN4 i DIN5. Oni puštaju napon na motore i zadovoljavaju pola uvjeta za uspostavu CAN komunikacije sa PLC-om. Prilikom paljenja sustava CANopen mrežom dolazi druga polovica uvjeta za uspostavu CAN komunikacije između PLC-a i kontrolera. Sama mreža radi na baudrate-u od 125 kb/s te može podržavati brzinu do 1 Mb/s.



Slika 14. Shema spoja PLC-a, motor kontrolera i motora

3.1. CANopen

Elementi CAN mreže su spojeni u seriji, tako da je potrebno na krajnjim elementima mreže moraju staviti otpornike od 120 oma, koji preko CAN-L i CAN-H žica međusobni spojeni u paraleli.



Slika 15. Shematski prikaz CAN open mreže

Rubni elementi ove CAN open mreže su PLC i motor kontroler za motor M1. Ranije je spomenuto da svaki motor kontroler ima na sebi sklopku za uključivanje takvog otpornika (Poglavlje 2.1.2.) na način da je jedan otpornik aktiviran na kontroleru, a drugi je ugrađen u konektor CAN kabela koji se spaja na PLC.

CAN mrežom se prenose od PLC do kontrolera podatci o zadanim željenim brzinama, pozicijama motora. Uz to, prenose se naredbe kojima kontroler zadaje novo dobivene pozicije i brzine na motor. Kontroler na PLC šalje podatke o trenutnom stanju položaja i brzina motora.

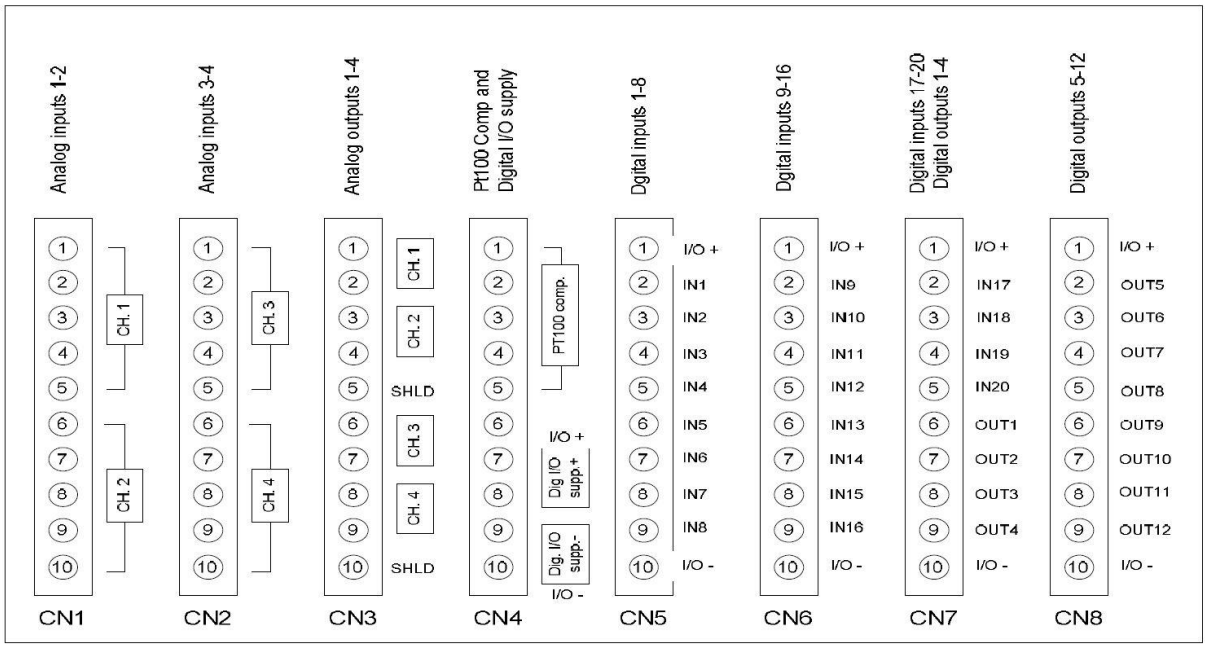
3.2. Digitalni I/O

Komunikacija digitalnim signalima ide preko CDPX-a na koji je montiran zasebni modul CDPX-EA-V1 s digitalnim i analognim ulazima i izlazima. Uz podizanje bitova DIN4 i DIN5 na digitalni input je spojena fizička sklopka u nuždi ili gljiva.

Naredbe poslane preko digitalnog I/O izvršavaju se pri pokretanju stroja, i u slučaju ako je došlo do deaktivacije gljive.

Slika 16. CDPX-EA-V1 modul

Naredna slika shematski je prikaz svih priključaka na CDPX-EA-V1. Modul ima osam kanala za priključke od kojih se na prva tri nalaze analogni ulazi/izlazi. Četvrti kanal je napajanje za digitalni I/O, dok se na preostalim kanalima nalaze digitalni ulazi i izlazi.



Slika 17. Priključci na CDPX-EA-V1

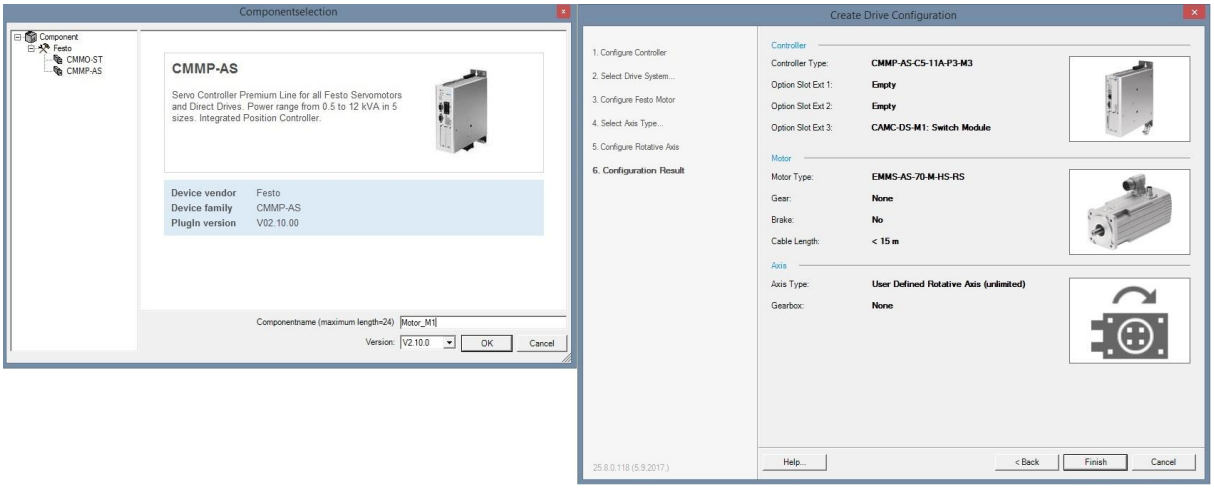
4. SOFTWARE I PROGRAMIRANJE

Programiranje PLC-a je napravljeno u programu CoDeSys. Za CECX korištena je verzija programa v2.3, dok je program za CDPX napravljen na verziji v3.5. Osim PLC potrebno je bilo programirati motor kontrolere, a to je napravljeno preko ranije spomenutog FCT programa.

Upravljači PLC cijelog sustava je CECX koji vrti glavni program upravljanja dok je CDPX pomoćni PLC i tu je samo kako bi postavio određene ulazne varijable.

4.1. FCT

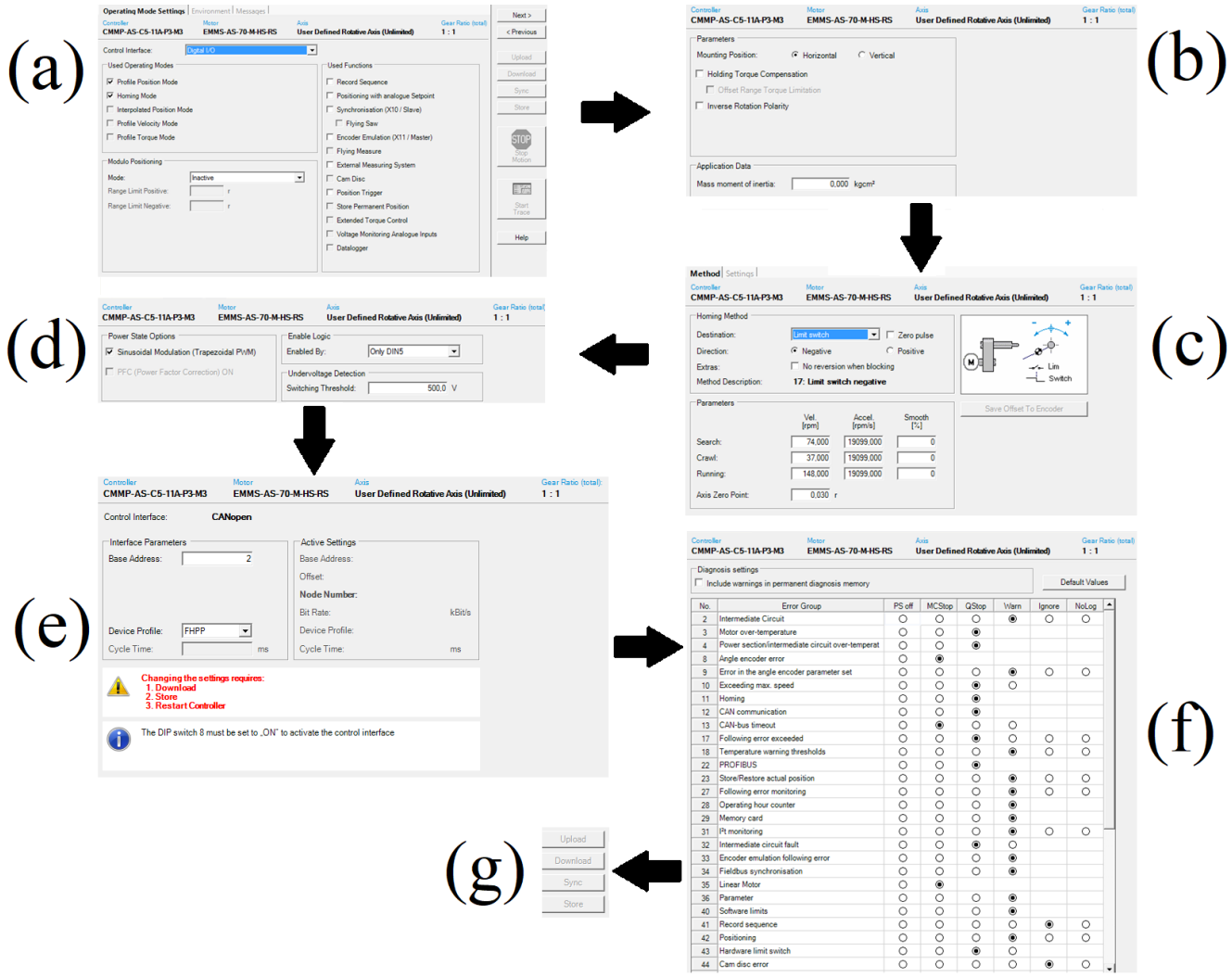
FCT ili Festo Configuration Tool je program za parametrizaciju svih serija FESTO-vih motor kontrolera. Otvaranjem novog predloška u FCT-u automatski se dodaje prva komponenta za koju korisnik prvo bira seriju kontrolera kojoj pripada komponenta koju će parametrizirati. Nakon odabira serije kreira se konfiguracija za odabranu komponentu, koja se sastoji od točnog definiranja kontrolera, motora i osi koju će isti pokretati. S prethodno napravljenim koracima završena je inicijalna postava komponente. Potrebno je naglasiti da se ovaj postupak ponavlja za svaku novo dodanu komponentu.



Slika 18. Početno sučelje kada odabiremo seriju kontrolera(lijevo) i zadnje korak prije inicijalnog postava komponente (desno)

Nakon inicijalne konfiguracije počinje parametrizacija. U ovom postupku korisnik prvo bira upravljačko sučelje za kontroler i njegove operativne modove . Naredni koraci sastoje se

od odabira: načina mrežne komunikacije (**a**), položaja motora (**b**), načina hominga (**c**), određivanje uvjeta za postavljanje kontrolera u 'enable' mod (**d**), konfiguracija CANopen portova (**e**) te postavljanje reakcija na pojavu greški(Error) u radu (**f**).



Slika 19. Glavni koraci u parametrizaciji kontrolera

Po završetku svih koraka parametrizacije spaja se sa FCT-om preko Etherneta na kontroler i onda se prvo prenesu(Download) zadani parametri u radu memoriju, te se zatim spreme (Store) (**g**) u trajnu memoriju kontrolera. Nakon parametrizacije potrebno je restartati kontroler, a za to se ide u alatnu traku pod component -> online -> restat controller, kako bi nove postavke kontrolera postale operativne.

4.2. CoDeSys

CoDeSys ili Controller Development System je razvojni program za izradu upravljačkih aplikacija za PLC-e prema industrijskom standardu IEC 61131-3. Program je razvila njemačka kompanija 3S-Smart Software Solutions iz grada Kemptena u Bavarskoj.

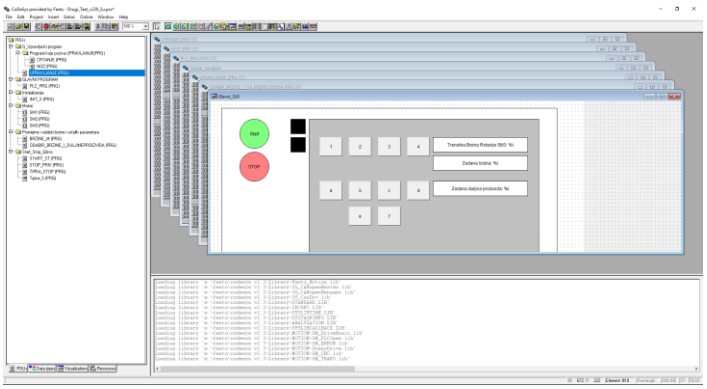
Zbog industrijskog standarda ICE 61131-3 CoDeSys podržava programiranje u pet programskih jezika:

1. IL (Instructon List)
2. ST (Structured text)
3. LD (Ledder diagram)
4. FBD (Function Block Dijagram)
5. SFC (Sequential Function Chart)

Uz ovih pet jezika omogućeno je i grafičko programsko sučelje CFC (Continous Function Chart) koje nije definirano IEC standardom. Kompajleri ovih programa pretvaraju njihove aplikaciju u binarni kod, koji se onda sprema na PLC.

Kako je CoSeSys napravljen po IEC standardu koriste ga svi proizvođači PLC-a, kao što su Simens, FESTO itd. Svaki proizvođač integrirao ga je u svoj programski paket, tako sa npr. FESTO-vim Codesys-om ne može programirati Simens-ov PLC i obrnuto.

Za izradu ovog rada korištene su dvije verzije Codesys-a zbog upotrebe dvaju PLC-ova od kojih se svaki morao programirati sa svojim verzijom, tako da je CECX programiran sa Codesys-om v2.3, a CDPX sa v3.5



Slika 20. Sučelje CoDeSys v2.3

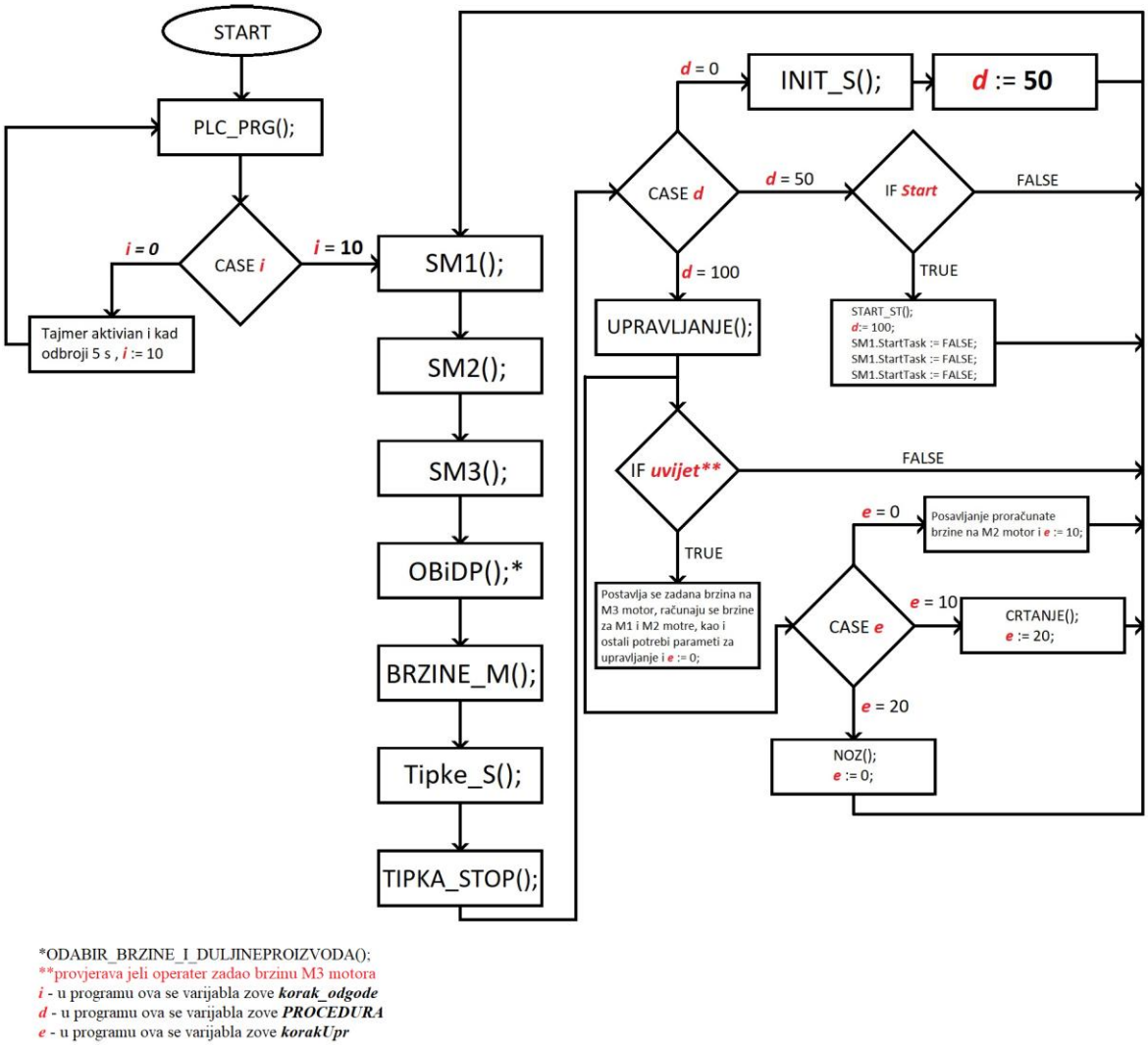
4.3. Rad glavnog programa i dijagram toka

Programiranje je napravljeno u CoDeSys-u i koristili su se sljedeći programski jezici za pisanje skripti: ST(Structured text), LD(Ledder diagram) i FBD(Function Block Dijagram). Program se sastoji od sljedećih četrnaest skripti. U narednoj tablici su navedene sve skripte i njihovi programski jezici.

Tablica 2. Skripte programa

Ime skripte	Programski jezik skripte	Opis
PLC_PRG();	ST(Structured text)	Glavni program
SM1();	FBD(Function Block Dijagram)	Blok za komunikaciju za SM1
SM2();	FBD(Function Block Dijagram)	Blok za komunikaciju za SM2
SM3();	FBD(Function Block Dijagram)	Blok za komunikaciju za SM
OBiDP();	ST(Structured text)	Odabir parametara za upravljanje
BRZINE_M();	ST(Structured text)	Zadaje brzinu na motore
Tipke_S();,	LD(Ledder diagram)	Omogućava rad tipki na GUI-u
STOP_PRIV();	ST(Structured text)	Zaustavlja sustav
TIPKA_STOP();	ST(Structured text)	Poziva skriptu STOP_PRIV();
INIT_S();	ST(Structured text)	Dovodi sustav u početno sanje nakon paljena
START_ST();	ST(Structured text)	Pokreće sustav
UPRAVLJANJE();	ST(Structured text)	Računa potrebne parametre upravljanja i pozva upravljačke skripte
CRTANJE();	ST(Structured text)	Upravlja položajem motora M2
NOZ();.	ST(Structured text)	Upravlja brzinom motora M1

Slika 21. prikazuje dijagram toka izvršavanja gore navedenih programa zajedno s uvjetima za njihovo pozivanje i aktivaciju.



Slika 21. Dijagram toka programa na PLC-u CECX-X-M1

Program kao cjelina radi na sljedeći način: prilikom paljenja sustava prvo se poziva PLC_PRG(); skripta i pali se tajmer sa odgodom od 5s; za vrijeme odbrojavanja tajmera drugi PLC u sustavu odradi svoj funkciju podizanja bitova na motor kontrolerima; po isteku vremena odgode počinju se pozivati i izvršavati skripte sljedećim redoslijedom: ();, SM1();, SM2();, SM3();, OBiDP();, BRZINE_M();, Tipke_S();, TIPKA_STOP(); i uz to je uvjet za poziv i izvršenje skripte INIT_S(); ispunjen. Od svih skripti koje se pozivaju pri paljenju sustava INIT_S(); je najvažnija jer dovodi sustav u radno spremno stanje.

Nakon što je sustav u radnom spremnom stanju čeka se operatera da preko GUI-a unese parametre i pokrene sustav tipkom start. Unošenjem parametara aktivira se skripta OBiDP();

koja sadrži vrijednosti proizvoda. Pokretanjem sustava poziva se skripta UPRAVLJANJE();. Ona prvo izračunava se preostale parametre upravljanja, na temelju unesenih te potom poziva skripte CRTANJE(); i NOZ(); koje izvršavaju upravljanje.

Skripta TIPKA_STOP(); se aktivira ako je operater na grafičkom sučelju pritisnuo tipku STOP. Ona provjerava stanje bita Pomoćni_stop iz skripte Tipke_S(); i ako je bit u logičkoj jedinici poziva se skripta STOP_PRIV();. Ova skripta zaustavlja sustav u trenutnom položaju i ne resetira ga. Tako sustav ponovnim pritiskom tipke START poziva skriptu START_ST(); i nastavlja sa radom iz položaja u kojem je zaustavljen.

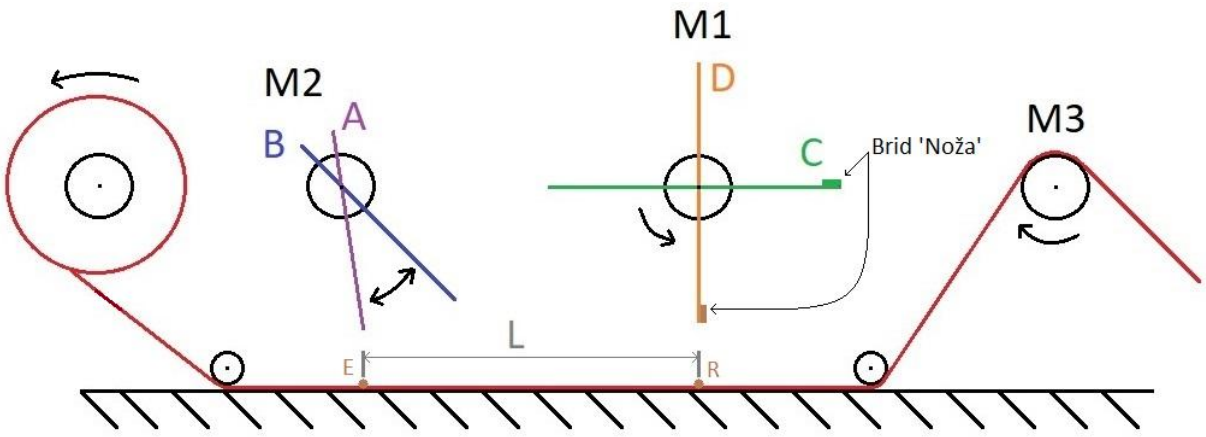
Preko TASK CONFIGURATION u CoDeSys-u postavlja se zadatak programa koji onda poziva skriptu. Ovdje je moguće dodati nekoliko zadataka sa pozivima različitih skripti. Uz to se definira prioritet zadatka i način njegovog izvođenja. Prioritet određuje koji se od zadataka prvi izvršava. Izvođenje može biti cikličko (cyclic), slobodno (freewheeling) i aktivirano nekim vanjskim ili unutarnjim događajem (npr. pokretanje neke funkcije unutar programa, aktivacija senzora). Za ovaj rad prioritet nije bitan jer postoji samo jedan zadatak, a način izvođenja je stavljen na „slobodan“ (freewheeling). Ovaj načina ponovo poziva skriptu po završetku njezinog izvršenja.

5. UPRAVLJANJE I REZULTATI

Prvim paljenjem makete motori se dovode u svoje startne pozicije, a zatim operater zadaje parametre upravljanja kao što su dužina proizvoda i brzina kretanja trake. Cijeli se sustav pokreće pritiskom na tipke start na upravljačkom sučelju. Svi upravljački proračuni i ostale operacije odvijaju se na temelju unesenih parametara.

5.1. Konceptualno rješenje sinkronizacije

Pokretanjem sustava prvo motor M3 treba postići zadanu maksimalnu brzinu ako bi se aktivirali algoritmi sinkronizacije i upravljanja. To je napravljeno na taj način zato jer je M3 vodeći motor sinkronizacije sustava. Naime, na temelju njegove brzine proračunava se brzina motora M1, a s obzirom na kutnu poziciju njegove rotacijske osi određuje se pozicija motora M3.



Slika 22. Skica sustava sa pozicijama motora M1 i M2

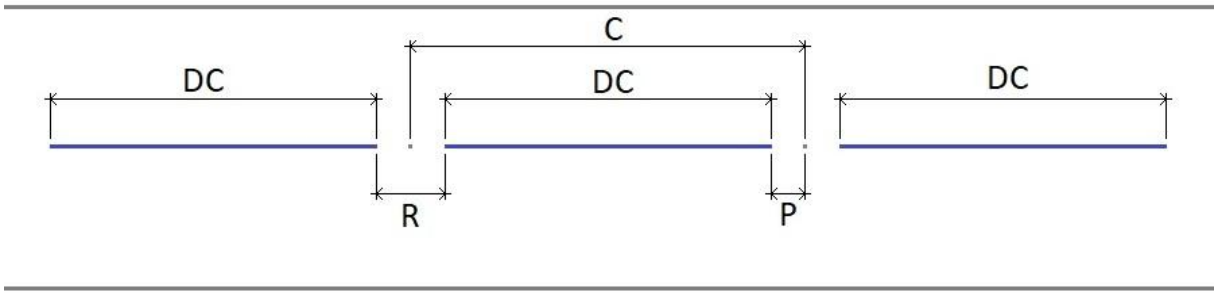
Gornja slika pokazuje koncept sinkronizacije sustava. Motor M3 se vrti konstantom brzinom. Poznata je udaljenost od točke zacrtavanja (E) predmeta do točke obilježavanja rezanja (R) folije na dužinu predmeta; ta udaljenost obilježena je sa 'L' i konstanta je. Na slici se vidi da motor M2 ima dvije pozicije A-crtanje i B-pauza, dok motor M1 ima startnu poziciju C i poziciju reza D.

Motor M2 periodički mijenja svoju poziciju s obzirom na trenutnu poziciju M3 motora. Za M1 se računa njegova konstantna brzina okretanja, jer kada se on jednom pokrene njegovo

gibanje ostaje nepromijenjeno sve dok operater ne zaustavi ili ugasi sustav. Bitno je naglasiti da se M1 ne počinje gibati prilikom pokretanja sustava, nego tek kada M3 motor pomakne papir za udaljenost N_d. Ovu udaljenost program računa nakon što operater unese sve potrebne parametre.

5.2. Sinkronizacija

Prethodno je opisan koncept sinkronizacije motora u sustavu, a ovdje se prikazuju matematičke formule i ostali dijelovi koji omogućavaju funkcionalnost tog koncepta.



Slika 23. Duljine proizvoda i razmaci među njima

Na gornjoj slici je skiciran prikaz izgleda papira nakon što se na njemu obilježe duljine proizvoda i pauze. Svaki proizvod je iste duljine (***DC***) kao i razmaci (***R***) među njima. Iz ovoga se definira dužina (***C***).

$$C = DC + R \quad \textbf{(1)}$$

Tablica 3. Vrijednosti DC i C za pet različitih slučajeva

DC (mm)	*R*(mm)	*C*(mm)	Okr. osi M3 za *C*	Inkr. iznos za *C*
55	14	69	1.25	81920
60	14	74	1.34	87818
65	14	79	1.44	94371
70	14	84	1.53	10027
75	14	89	1.62	106168
80	14	94	1.71	112066

U Tablici 4 dane su brzina M3 motora u tri kategorije. Prva kategorija je brzina zadana preko CoDeSys-a, druga je ta brzina očitana iz FCT-a i treća je izračunata obodna brzina M3.

Obodna brzina računa se prema formuli:

$$v_{0,z} = \frac{rn\pi}{30} \quad \textbf{(2)}$$

Uvrštavanjem sljedećih vrijednosti n = 1 o / min i r = 8.75 mm dobiva se obodna brzina motora M3 koja se uzima za referentnu vrijednost brzine i ima oznaku $v_{0,r}$. Obodna brzina je ujedno i brzina pomicanja papira.

Tablica 4. Vrijednosti brzina M3 motora

Brzina u CoDeSys-u $v_{co,z}$ (ink/s)	Okretaji ***n*** (o/min)	Obodna brzina $v_{o,z}$ (mm/s)
4167	1	0.9163
5000	1.2	1.0996
10000	2.4	2.1991
15000	3.6	3.2987
20000	4.8	4.3982

Referentni podatci tablice 4 su u prvom retku i vrijednosti u prvom stupcu su zadane, a ostali su izračunati preko sljedećeg omjera:

$$\frac{v_{co,r}}{v_{co,z}} = \frac{v_{o,r}}{v_{o,z}} \quad \textbf{(3)}$$

$$v_{o,z} = v_{o,r}\frac{v_{co,z}}{v_{co,r}} \quad \textbf{(4)}$$

Sad kada je poznata obodna brzina M3 motora, može se izračunati vrijeme potrebno da se M1 motor okrene za jedan puni krug, a u kojem M3 pomakne papir za udaljenost C:

$$t_p = \frac{C}{v_{o,z}} \quad \textbf{(5)}$$

Uvrštavanjem (4) u (5) dobiva se izraz za izračun vremena koja koristiti referentne podatke iz tablice 2, zadanu brzinu ($v_{co,z}$) i konstantu udaljenosti između dviju točki rezanja (C).

$$t_p = \frac{C \times v_{co,r}}{v_{o,r} \times v_{co,z}} \quad \textbf{(6)}$$

$$\omega_p = \frac{\varphi}{t_p} \quad \textbf{(7)}$$

Jednadžba (6) se potom uvrštava u jednadžbu za kutnu brzinu (7) i dobivamo sljedeće:

$$\omega_p = \varphi\frac{v_{o,r} \times v_{co,z}}{C \times v_{co,r}} \quad \textbf{(8)}$$

$$\varphi = \alpha \frac{\pi}{180} \quad \textbf{(9)}$$

Pri čemu su:

- φ - kut u radijanima
- α - kut u stupnjevima

Uvrštavanjem (9) u (8) dobiva se:

$$\omega_p = \alpha \frac{\pi \times v_{o,r} \times v_{co,z}}{180 \times C \times v_{co,r}} \quad \textbf{(10)}$$

Gornjom formulom se izračuna konstantna kutna brzina motora M1. Nakon toga ju je potrebno prevesti u jedinicu brzine koju koristi CoDeSys. Kako bi se to napravilo potrebno je prvo postaviti omjer kutnih brzina u odnosu na brzine CoDeSys-a, a to je prikazano sljedećim izrazom gdje su ω_r i $v_{co,r}$ konstante te iznose ω_r= 0.105 s^{-1} i $v_{co,r}$= 4167 ink/s:

$$\frac{\omega_r}{\omega_p} = \frac{v_{co,r}}{v_{co,p}} \quad \textbf{(11)}$$

$$v_{co,p} = v_{co,r} \frac{\omega_p}{\omega_r} \quad \textbf{(12)}$$

Uvrsti se (10) u (12) i dobije se:

$$v_{co,p} = \frac{\alpha \times \pi \times v_{o,r} \times v_{co,z}}{180 \times C \times \omega_r} \quad \textbf{(13)}$$

Formulom (13) računa brzina M1 motora za kut α = 360˚. Ona je konstantna tijekom rada i mijenja se jedino ako dođe do promijene brzine M3 motora.

M1 se ne počinje gibati pokretanjem sustava već čeka da M3 pomakne papirnu traku za udaljenost N_d, koja se dobiva iz razlike:

$$N_d = L_k - d \quad \textbf{(14)}$$

Formula za duljinu *d* izvedena je iz jednadžbe (13):

$$d = \frac{\alpha \times \pi \times v_{o,r} \times v_{co,z}}{180 \times v_{co,p} \times \omega_r} \quad \textbf{(15)}$$

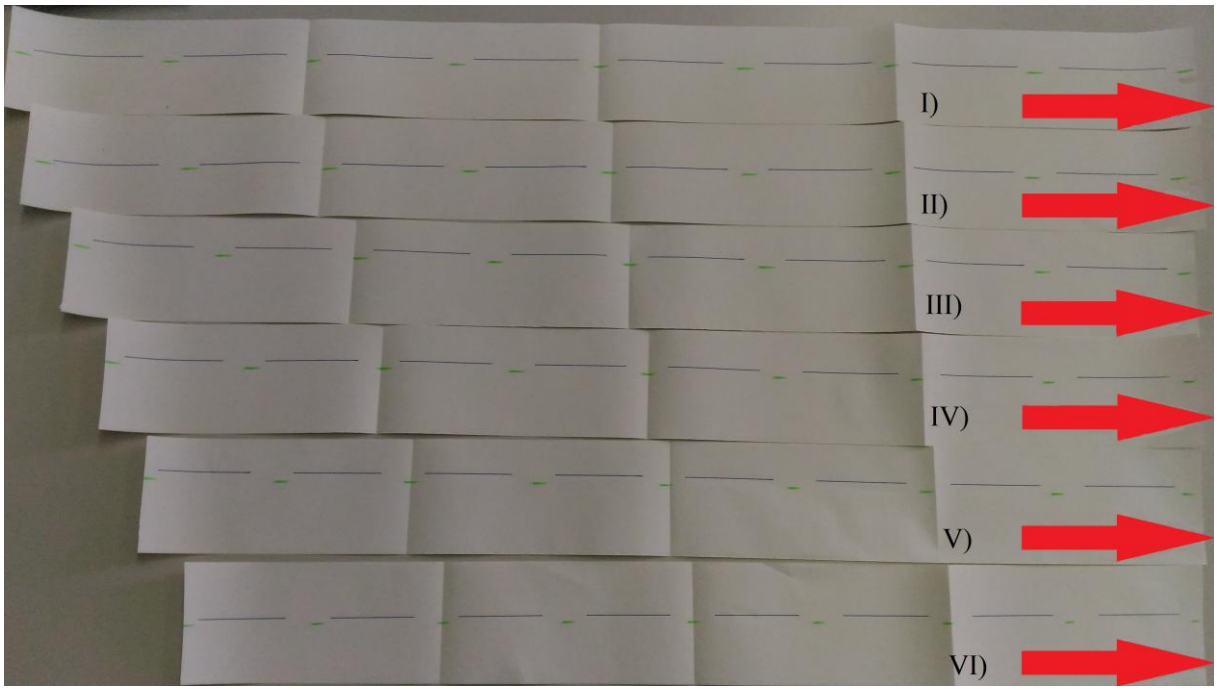
Duljina *d* je pomak papirne trake za koji motor M1 napravi kutni pomak od početne do pozicije rezanja sa brzinom izračunatom u jednadžbi (13), taj kut u ovom slučaju iznosi 270˚. Nakon što se izračuna *d*, može izračunati N_d čija je vrijednost u milimetrima i potrebno ju je prevesti u jedinicu koju koristi CoDeSys formulom:

$$N_{di} = \frac{N_d}{55} \times 65536 \quad \textbf{(16)}$$

Cijeli proračun se provodi prije početka pokretanja sustava i najvažniji izračunati podatci su: brzina M1 motora ($v_{co,p}$) i duljina (N_{di}) za koju M3 mora pomaknuti papirnu traku kako bi se pokrenuo M1 motor.

5.3. Rezultati i komentari ispitivanja

Rezultati testiranja rada sustava napravljeni su pri brzini vodećeg motora od 4.4 mm/s počevši sa duljinom proizvoda od 55 mm do 80 mm s inkrementalnim povećanjem po 5 mm do najveće duljine proizvoda.



Slika 24. Rezultati ispitivanja rada sustava

Tablica 5. Parametri za slučajeve ispitivanja na slici 24

Oznaka	DC(mm)	R(mm)	$v_{co,z}$ (ink/s)	$v_{o,z}$ (mm/s)
I)	55	30	20000	4.4
II)	60	30	20000	4.4
III)	65	30	20000	4.4

IV)	70	30	20000	4.4
V)	75	30	20000	4.4
VI)	80	30	20000	4.4

Slika 24 prikaz je dobivenih rezultata, a crvene strelice na njoj označavaju smjer povlačenja papira. Plave linije prikazuju duljinu proizvoda, a zelene mjesto reza noža. Duljina plavih linija i razmaka među njima određena je brojem okretaja vodećeg motora sustava, a razmaci između rezova određeni su brzinom motora za nož.

Plave linije i razmake, crta motor M2 koji ima dva fiksna položaja. Prvi je položaj crtanja, a drugi položaj pauze. Položaji se izmjenjuju tijekom rada sustava, a vrijeme koje M2 provede u svakom položaju određeno je brojem okretaja M3 motora. Kao primjer možemo reći da je motor M2 u poziciji crtanja sve dok M3 ne napravi jedan puni krug te se onda pomakne u poziciju pauze gdje čeka da M3 napravi pola kruga.

Brzina okretaja M1 je uvijek konstanta i računa se nakon unošenju upravljačkih parametara u grafičko sučelje sustava. M1 motor se aktivira i održava zadanu brzinu do zaustavljanja sustava, vrijeme njegove aktivacije određeno je pozicijom M3 motora.

Sa slike 24 se vidi da dolazi do faznog pomaka reza (zelene linije) u odnosu na plave linije. Ovo se javlja zbog nepreciznog crtanja plavih linija, koje ispadaju veća od zadane dužine u rasponu od 0.5 do 1 mm. Pogreška crtanja se javlja pri svim brzinama pokretanja sustava. Jedina je razlika u tome da je pri manjim brzinama sustava pogreška manja i primjetna tek pri dužem radu sustava. Uzrok ove fazne pogreške se pridodaje nemogućnosti povećanja brzine mrežne komunikacije.

Zašto se cijelo vrijeme spominje brzina komunikacije kao glavni problem? Kako bi se na to odgovorilo prije svega treba konstatirati da se vrijednosti parametara, koji se povratnom vezom šalju nazad na PLC, uspoređuju sa uvjetom isključivo preko nejednakosti, nikad preko jednakosti. Ako uvjet treba biti istinit za eksplicitnu vrijednost parametra, onda je potrebno provjeravati da iznos parametra upada u interval oko te vrijednosti.

Ovakav pristup se koristi pri programiranju PLC zato jer se mora uzeti u obzir brzina mreže kojom se podatci izmjenjuju. Što je veća brzina to se više informacija izmjeni u kraćem vremenu pa intervali za eksplicitne vrijednosti mogu biti manji. U protivnom, ako je brzina mreže neadekvatna za određeni interval može se dogoditi da vrijednost parametra preskoči

raspon intervala što uzrokuje neispunjenje uvjeta, a to dovodi do nepokretanja neke fizičke komponente, neaktivacije senzora ili neke druge radnje. Iz prethodnog proizlazi da se pri većim brzinama mreže mogu stavljati manji intervali vrijednosti, a pri malim brzinama mreže ti intervali rastu.

Sad se na sve to nadoda brzina rada sustava, koja dodaje još jednu razinu složenosti. Naime, ako je brzina mrežne komunikacije za dani sustav maksimalna onda brzina rada sustava malo utječe na iznose intervala. Međutim, ako je brzina komunikacije minimalna stvari su obrnute. Kako ne bi došlo do toga da vrijednost preskoči interval, povećanjem brzine rada sustava dolazi do povećanja duljine intervala, što onda dovodi do unošenja sve veće pogreške.

Sustav napravljen za potrebe ovog rada, na principu intervalnih vrijednosti određuje poziciju M2 motora. Promjena njegove pozicije ovisi o dvije razlike. Apsolutne vrijednosti tih dviju razlika moraju biti manje od neke vrijednosti ***b*** i provjeravaju se naizmjenično; dok se ne ispuni uvjet prve razlike, druga razlika se ne računa i obrnuto. Kako mreža ne radi na brzini većoj od 125 kb/s za sve radne brzine M3 veće od 4.4 mm/s potrebno je povećati raspon intervala obiju razlika. Tim dolazi do povećanja pogreške pri crtanju linija te to dovodi do ranijeg pojavljivanja faznog pomak pri rezanju.

Brzina komunikacije ne utječe puno na M1 motor jer za njega postoji samo naredba početak rada konstantnom brzinom. Jedino što bi se moglo dogoditi da M1 malo kasni s pokretanjem zbog uvjeta koji ga aktivira, ali da se to i dogodi, fazni pomak koji se tom greškom unosi je kompenziran razmakom među proizvodima.

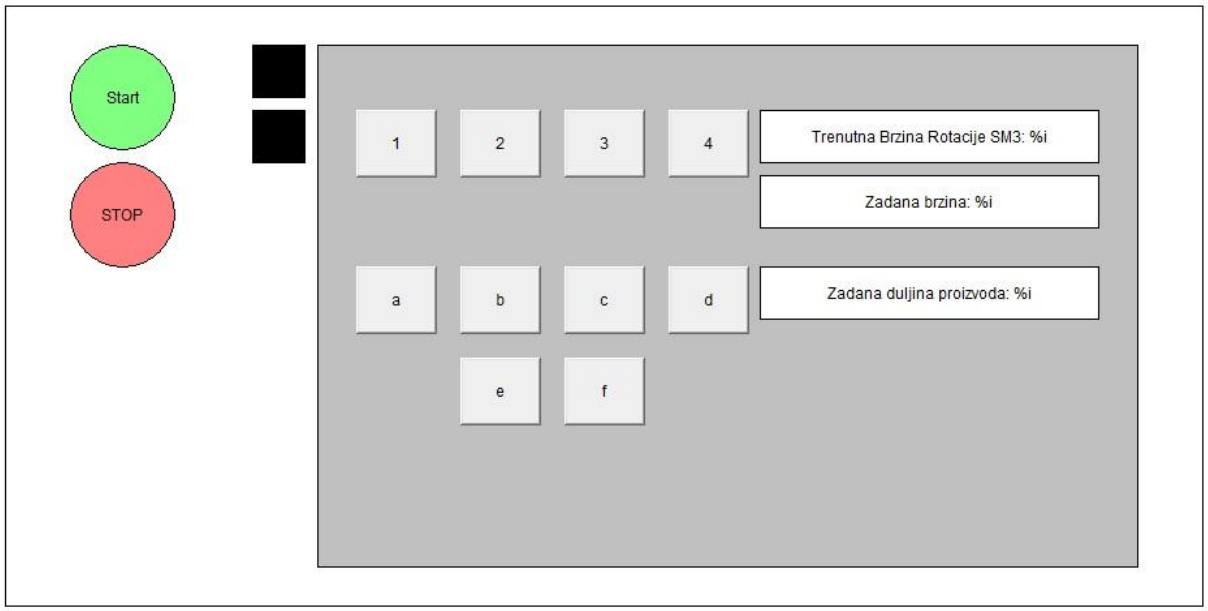
6. GRAFIČKO SUČELJE ZA UPRAVLJANJE

Svaki upravljački program treba imati grafičko sučelje (GUI) preko kojeg operater upravlja sustavom. GUI je most koji povezuje operatera i sustav na način da preko jednostavnih naredbi izvršava upravljanje i regulaciju sustava. Kod izrade GUI-a potrebno je obratiti pažnju na jednostavnost dizajna,a broj unosa potrebnih podataka treba biti minimalan. Ovakav dizajn za industrijske aplikacije je bitan jer pokretanje proizvodnih postupaka treba biti brzo i efikasno. Time se ujedno smanjuje potreba za dugom obukom operatera.

Broj podatka za unos je minimalan, ali se na temelju njih u pozadini računaju svi ostali potrebni podatci ili su uneseni podatci šifre za predefinirane skupine parametara koje sustav dohvaća kad je unesena njihova šifra.

GUI može i ne mora sadržavati tipke za pokretanje, zaustavljanje sustava. Ove tipke su u nekim slučajevima fizička tipkala i sklopke koje se nalaze pored ekrana za GUI. Jedina uvijek fizička sklopka je ona za zaustavljanje sustava u nuždi ili gljiva.

Na slici 25 prikazan je GUI napravljen za potrebe ovog diplomskog rada. Pri njegovoj izradi korišteni su gore opisani principi. Na lijevoj strani GUI-a vide se tipke: Start i Stop. Dva crna kvadrata su indikatori koji obavještavaju u kakvom je stanju sustav. Prvi indikator u stupcu svijetli zeleno nakon pritiska tipke Start i daje označava da sustav aktivan. Drugi svijetli narančasto kada se sustav zaustavi sa Stop tipkom. Desna strana GUI-a služi za unos potrebnih parametara. Unose se dva parametra. Prvi je brzina, koja se može mijenjati tijekom rada sustava. Onda se namješta tipkama od 1 do 4. Drugi parametar je duljina proizvoda koja se unosi tipkama od ***a*** do ***f***, duljina proizvoda se ne može mijenjati tijekom rada sustava. Ona se zadaje na početku rada i ako se želi mijenjati potrebno je sustav resetirati. Lijevo od tipki su tri prozora koji operateru daju informaciju o podatcima koje je zadao i uz to prikazuju trenutnu vrijednost gibanja papira.



Slika 25. Upravljačko sučelje (GUI)

7. ZAKLJUČAK

U ovom diplomskom radu prvo je dizajniran i konstruiran eksperimentalni postav koji simulira rad Flow pack stroja za pakiranje. Dizajn i konstrukcija napravljeni su u Catia V5R19 programu. Postav se pokreće trima FESTO-vim motorima oznaka EMMS-AS, a njima se upravlja pomoću triju motor kontrolera i dva PLC-a. Sustav je sinkroniziran, što ovdje znači da jedan motor vodi druga dva. U ovom slučaju vodeći motor ima oznaku M3 i povlači papirnu traku, a od preostalih prvi s oznakom M2 crta linije koje predstavljaju duljine proizvoda i treći sa oznakom M1 obilježava mjesta između zacrtanih linija.

Postav simulira rad Flow pack stroja na bazičnoj razini, ali su performanse sustava slabe. Glavna mana sustava je njegova brzina rada. M3 motor se ne smije pokrenuti obodnom brzinom većom od 4.4 mm/s.

Problem brzine proizlazi iz problema s opremom, a odnosi se na nemogućnost povećanja brzine CAN komunikacije. Baudrate CAN mreže postavljen je na 125 kb/s, što je najsporija moguća brzina. Za sve baudrate-ove veće od navedenog komunikacija između PLC-a i motor kontrolera nije se mogla uspostaviti.

Spora komunikacija znači neadekvatnu izmjenu podataka, koja rezultira gubitkom relevantnih vrijednosti istih za koje bi program na PLC-u ispunio uvjete za promjenu stanja sustava ili aktivirao neku drugu funkciju programa. Iz tog razloga je kod ovog sustava obodna radna brzina M3 motora ograničena na 4.4 mm/s jer kod većih brzina dolazi do gubitka, odnosno preskakanja vrijednosti podataka te M2 zapinje u jednom od svoja dva položaja i tu ostaje.

Zbog spore komunikacije javlja se nepreciznost crtanja linija sa M2 motorom. Ona je spomenuta u prethodnom poglavlju i varira između 0.5 do 1 mm u duljini te izaziva fazni pomak koji postaje primjetan nakon izvjesnog vremena rada sustava.

Prvo poboljšanje sustava bio bi ispravak tehničkih pogreški te omogućavanje većeg baudrate-a. Ovime bi se otklonili i smanjili trenutni problemi sustava. Sljedeća stvar koja bi se mogla uvesti je senzor koji detektira dolazak zacrtane linije blizu mjesta rezanja te se na temelju njegove vrijednosti dodatno regulira brzina motora M1 kako bi se dobila točnija pozicija mjesta reza. To bi efektivno riješilo problem faznog pomaka.

Daljnja nadogradnja ovog bazičnog sustava bi bila izrada vjerodostojnije makete, koja bolje simulira rad Flow pack-a. Osim izrade nove makete bilo bi dobro koristiti programski

dodatak softmotion za CoDeSys. Softmotion povećava opseg funkcija i načina programiranja s bazične razine logike na napredniju razinu.

Ovaj rad daje dobar uvod u područje sinkronog upravljanja sustavima preko PLC-a. Nastavak rada u ovom području je preporučljiv i bitan jer može imati izravan utjecaj na industriju inovativnim i efikasnijim rješenjima upravljanja industrijskim pogonima.

LITERATURA

[1] https://www.festo.com/net/da_dk/SupportPortal/default.aspx?tab=3&q=553853, prestupljeno 25.10.2016.

[2] https://www.festo.com/cat/en-gb_gb/data/doc_engb/PDF/EN/CECX_EN.PDF, prestupljeno 25.10.2016.

[3] https://www.festo.com/net/SupportPortal/Files/326387/CDPX_2013-03_8004848g1.pdf, prestupljeno 11.4.2016.

[4] https://www.festo.com/cat/en-gb_gb/data/doc_ENUS/PDF/US/CDPX_ENUS.PDF, prestupljeno 11.4.2016.

[5] https://www.festo.com/cat/en-gb_gb/data/doc_ENUS/PDF/US/EMMS-AS_ENUS.PDF, prestupljeno 11.4.2016.

[6] https://www.festo.com/net/SupportPortal/Files/380659/CMMP-AS-M3-HW_2012-03_760322g1.pdf, prestupljeno 9.2.2017.

[7] https://www.festo.com/cat/sk_sk/data/doc_engb/PDF/EN/CMMP-AS-M3_EN.PDF, prestupljeno 9.2.2017.

[8] http://marie-www.ee.pw.edu.pl/~purap/PLC/manuals/m07590333_00000000_1en.pdf, prestupljeno 10.10.2017.

[9] https://www.festo.com/net/SupportPortal/Files/326384/CDPX_2013-03a_8004844d5.pdf, prestupljeno 14.12.2017

[10] https://www.festo.com/net/SupportPortal/Files/326671/CMMP-AS-M3_M0-C-CO_2013-04a_8022083g1.pdf, prestupljeno 10.1.2018.

[11] https://www.festo.com/net/SupportPortal/Files/326791/CMMP-AS-M3_M0-C-HP_2013-04a_8022075g1.pdf, prestupljeno 10.1.2018.

[12] https://support.crosscontrol.com/sites/default/files/documentation/IO%20Controllers/CrossFire%20MX1B%20CoDeSys/Manuals/CANopen%20for%203S%20Runtime%20Systems%20V2_3_5_0.pdf, 31.1.2018.

PRILOZI

I. Programski kod CoDeSys v2.3

II. Tehnička dokumentacija

III. CD-R disc

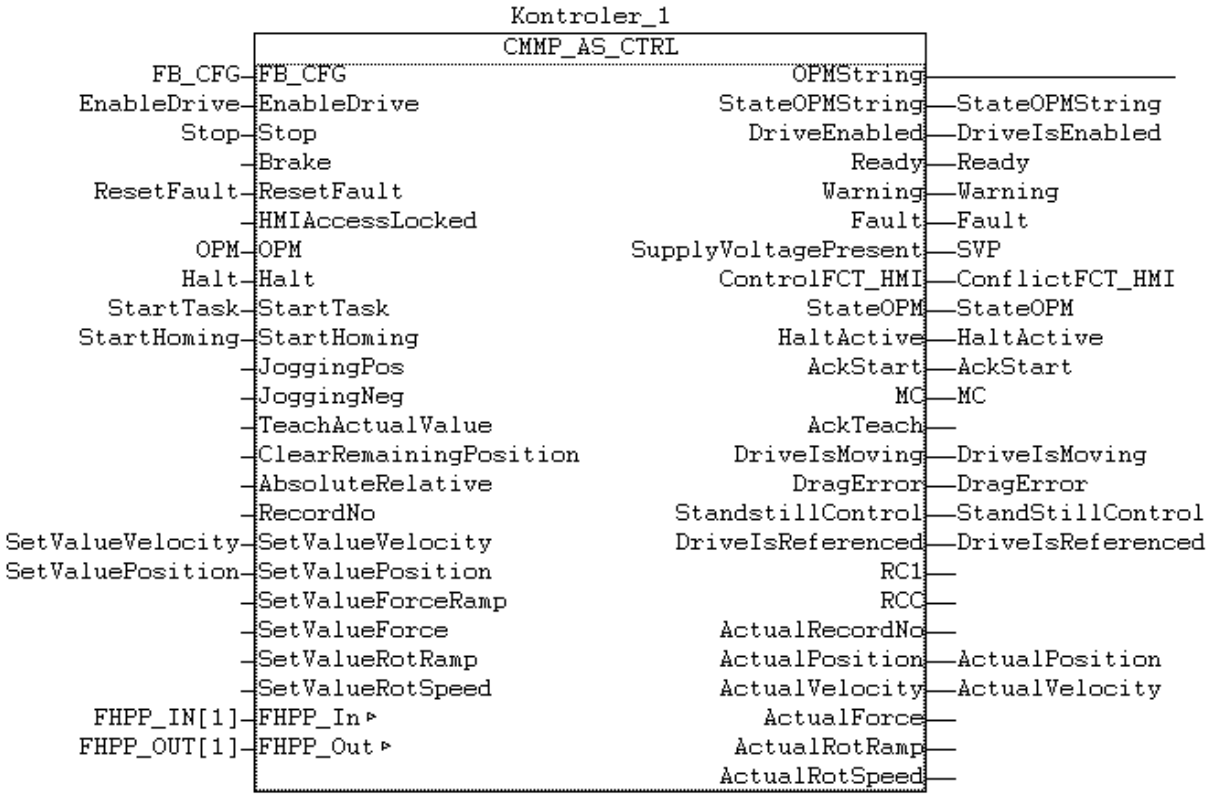
I. Programski kod CoDeSys

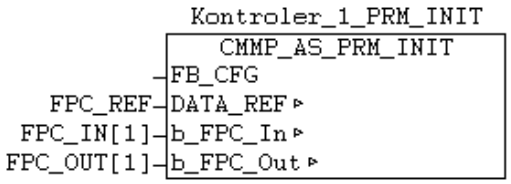
I.1. CoDeSys v2.3

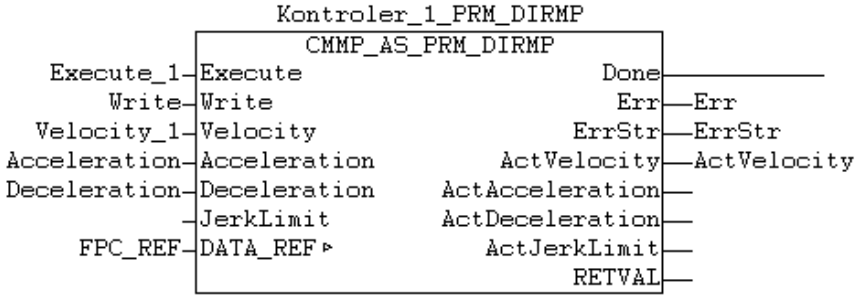
I.1.1. PLC_PRG();

```
CASE korak_odgode OF
0: odgoda(IN := TRUE, PT := t#5s);
        IF odgoda.Q THEN
                korak_odgode := 10;
        END_IF
10:     SM1();
        SM2();
        SM3();

        ODABIR_BRZINE_I_DULJINE_PROIZVODA();
        BRZINE_M();
        Tipke_S();
        TIPKA_STOP();
        CASE PROCEDURA OF
                0 :     INIT_S();
                50 : IF Start THEN
                                START_ST();
                                PROCEDURA := 100;
                                SM1.StartTask := SM1.StartTask := SM1.StartTask :=
FALSE;
                END_IF
                  100 : UPRAVLJANJE();

        END_CASE
END_CASE
```

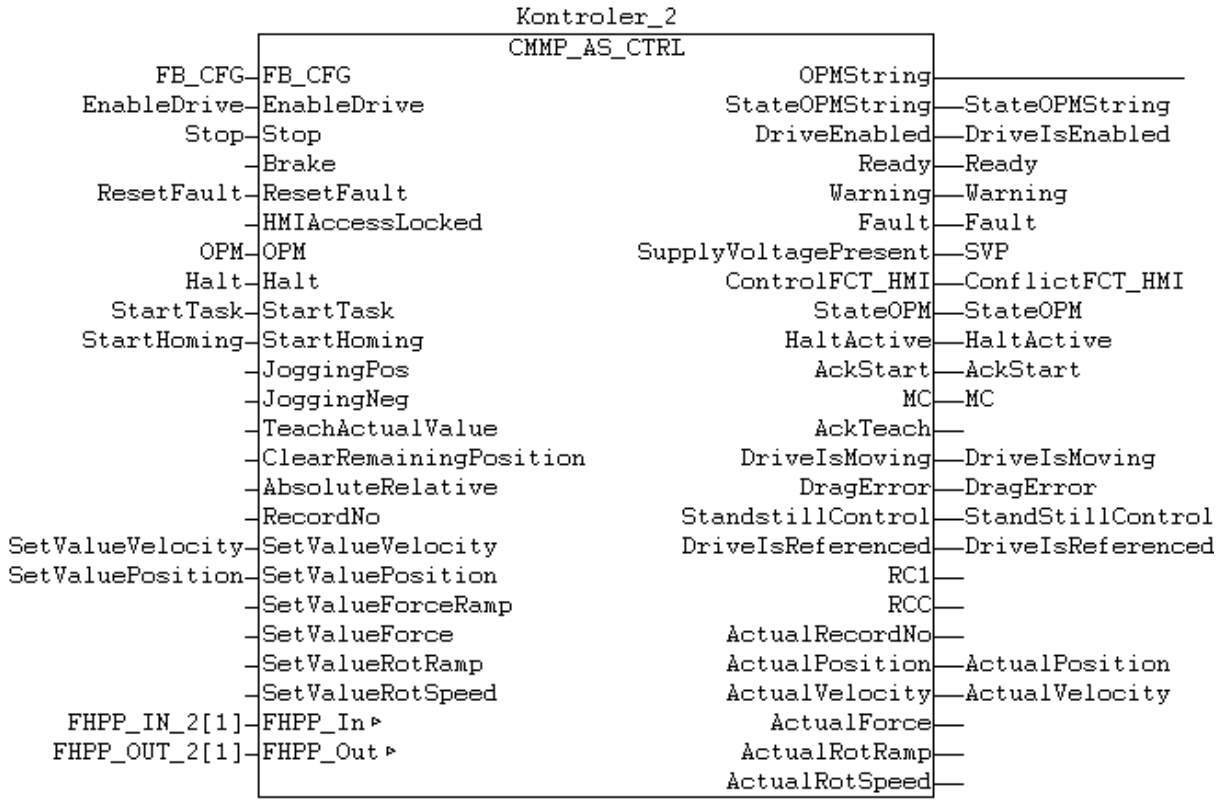
I.1.2. SM1();

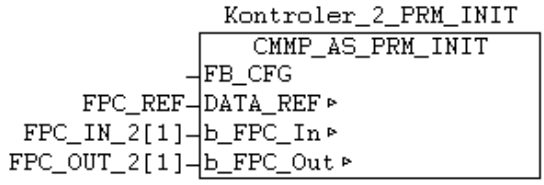

Kontroler_1
CMMP_AS_CTRL
FB_CFG
EnableDrive
Stop
Brake
ResetFault
HMIAccessLocked
OPM
Halt
StartTask
StartHoming
JoggingPos
JoggingNeg
TeachActualValue
ClearRemainingPosition
AbsoluteRelative
RecordNo
SetValueVelocity
SetValuePosition
SetValueForceRamp
SetValueForce
SetValueRotRamp
SetValueRotSpeed
FHPP_IN[1]
FHPP_OUT[1]
OPMString
StateOPMString
DriveEnabled
Ready
Warning
Fault
SupplyVoltagePresent
ControlFCT_HMI
StateOPM
HaltActive
AckStart
MC
AckTeach
DriveIsMoving
DragError
StandstillControl
DriveIsReferenced
RC1
RCC
ActualRecordNo
ActualPosition
ActualVelocity
ActualForce
ActualRotRamp
ActualRotSpeed
DriveIsEnabled
ConflictFCT_HMI
SVP

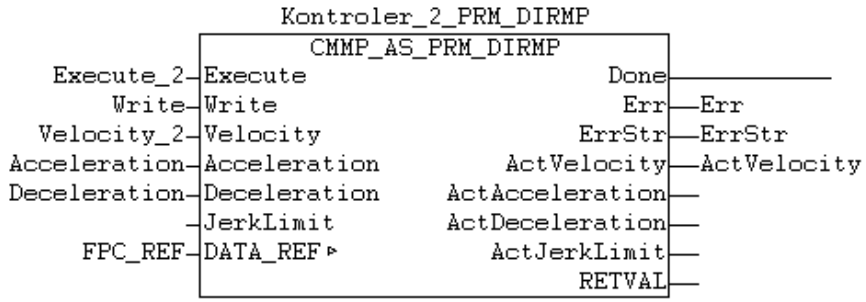

Kontroler_1_PRM_INIT
CMMP_AS_PRM_INIT
FB_CFG
DATA_REF
b_FPC_In
b_FPC_Out
FPC_REF
FPC_IN[1]
FPC_OUT[1]


Kontroler_1_PRM_DIRMP
CMMP_AS_PRM_DIRMP
Execute
Write
Velocity
Acceleration
Deceleration
JerkLimit
DATA_REF
Done
Err
ErrStr
ActVelocity
ActAcceleration
ActDeceleration
ActJerkLimit
RETVAL
Execute_1
Velocity_1
FPC_REF

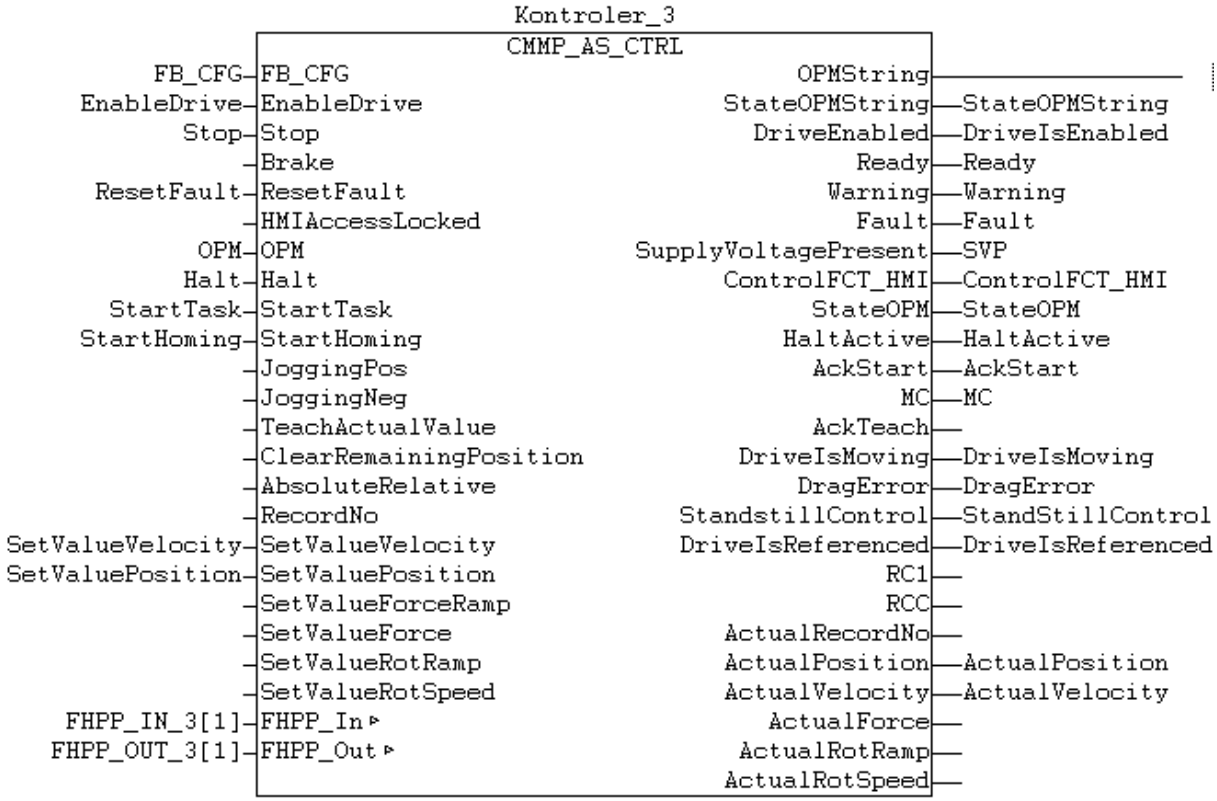
I.1.3. SM2();

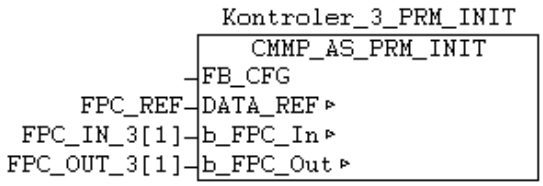

Kontroler_2
CMMP_AS_CTRL
FB_CFG
EnableDrive
Stop
Brake
ResetFault
HMIAccessLocked
OPM
Halt
StartTask
StartHoming
JoggingPos
JoggingNeg
TeachActualValue
ClearRemainingPosition
AbsoluteRelative
RecordNo
SetValueVelocity
SetValuePosition
SetValueForceRamp
SetValueForce
SetValueRotRamp
SetValueRotSpeed
FHPP_In
FHPP_Out
FHPP_IN_2[1]
FHPP_OUT_2[1]
OPMString
StateOPMString
DriveEnabled
Ready
Warning
Fault
SupplyVoltagePresent
ControlFCT_HMI
StateOPM
HaltActive
AckStart
MC
AckTeach
DriveIsMoving
DragError
StandstillControl
DriveIsReferenced
RC1
RCC
ActualRecordNo
ActualPosition
ActualVelocity
ActualForce
ActualRotRamp
ActualRotSpeed
DriveIsEnabled
SVP
ConflictFCT_HMI

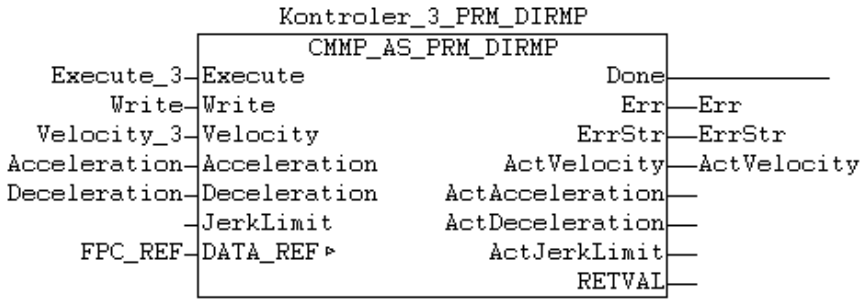

Kontroler_2_PRM_INIT
CMMP_AS_PRM_INIT
FB_CFG
DATA_REF
b_FPC_In
b_FPC_Out
FPC_REF
FPC_IN_2[1]
FPC_OUT_2[1]


Kontroler_2_PRM_DIRMP
CMMP_AS_PRM_DIRMP
Execute
Write
Velocity
Acceleration
Deceleration
JerkLimit
DATA_REF
Done
Err
ErrStr
ActVelocity
ActAcceleration
ActDeceleration
ActJerkLimit
RETVAL
Execute_2
Velocity_2
FPC_REF

I.1.4. SM3();


Kontroler_3
CMMP_AS_CTRL
FB_CFG
EnableDrive
Stop
Brake
ResetFault
HMIAccessLocked
OPM
Halt
StartTask
StartHoming
JoggingPos
JoggingNeg
TeachActualValue
ClearRemainingPosition
AbsoluteRelative
RecordNo
SetValueVelocity
SetValuePosition
SetValueForceRamp
SetValueForce
SetValueRotRamp
SetValueRotSpeed
FHPP_IN_3[1]
FHPP_OUT_3[1]
OPMString
StateOPMString
DriveEnabled
Ready
Warning
Fault
SupplyVoltagePresent
ControlFCT_HMI
StateOPM
HaltActive
AckStart
MC
AckTeach
DriveIsMoving
DragError
StandstillControl
DriveIsReferenced
RC1
RCC
ActualRecordNo
ActualPosition
ActualVelocity
ActualForce
ActualRotRamp
ActualRotSpeed


Kontroler_3_PRM_INIT
CMMP_AS_PRM_INIT
FB_CFG
FPC_REF
DATA_REF
FPC_IN_3[1]
b_FPC_In
FPC_OUT_3[1]
b_FPC_Out


Kontroler_3_PRM_DIRMP
CMMP_AS_PRM_DIRMP
Execute_3
Execute
Write
Write
Velocity_3
Velocity
Acceleration
Acceleration
Deceleration
Deceleration
JerkLimit
FPC_REF
DATA_REF
Done
Err
ErrStr
ActVelocity
ActAcceleration
ActDeceleration
ActJerkLimit
RETVAL

I.1.5. ODABIR_BRZINE_I_DULJINE_PROIZVODA(); skraćeno OBiDP();

```
IF I = TRUE THEN
        NovaBrzina_M3 := 5000;
END_IF
IF II = TRUE THEN
        NovaBrzina_M3 := 10000;
END_IF
IF III = TRUE THEN
        NovaBrzina_M3 := 15000;
END_IF
IF IV = TRUE THEN
        NovaBrzina_M3 := 20000;
END_IF

IF aI = TRUE THEN
        DC := 55;
        DCi := 65536;
        Cinkr := 101283;
        nevidlji := TRUE;
END_IF
IF bI = TRUE THEN
        DC := 60;
        DCi := 71494;
        Cinkr := 107241;
        nevidlji := TRUE;
END_IF
IF cI = TRUE THEN
        DC := 65;
        DCi := 77452;
        Cinkr := 113199;
        nevidlji := TRUE;
END_IF

IF dI = TRUE THEN
        DC := 70;
        DCi := 83409;
        Cinkr := 119156;
        nevidlji := TRUE;
END_IF
IF eI = TRUE THEN
        DC := 75;
        DCi := 89367;
        Cinkr := 125114;
        nevidlji := TRUE;
END_IF
```

```
IF fI = TRUE THEN
        DC := 80;
        DCi := 95325;
        Cinkr := 131072;
        nevidlji := TRUE;
END_IF
```

I.1.6. BRZINE_M();

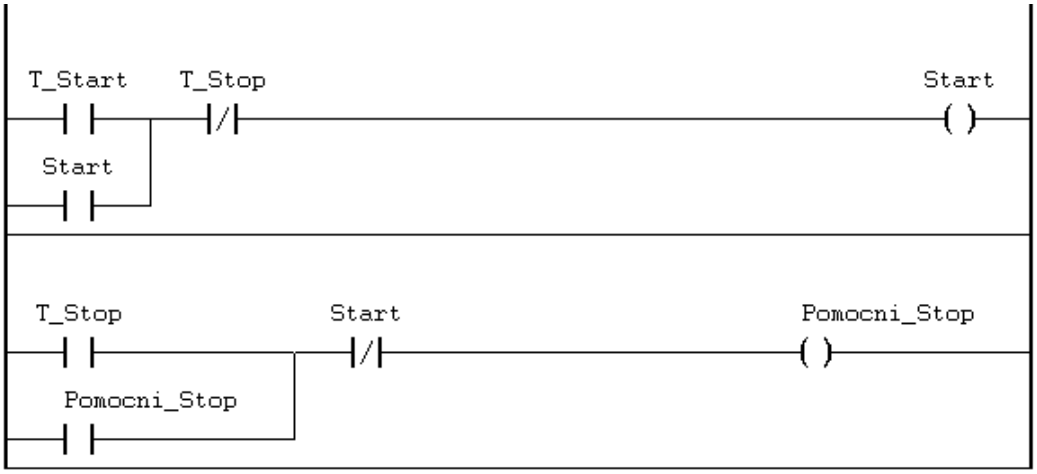
I.1.7.

```
(*Promjena brzine za kontroler SM1*)
IF M1.Promjena_brzine_type = FALSE THEN
        CASE M1.korak_type OF
                0 : SM1.StartTask := FALSE;
                        SM1.Execute_1 := FALSE;
                        SM1.Write := TRUE;
                        SM1.Velocity_1 := M1.brzina_type;
                        IF (SM1.AckStart = FALSE) AND
(SM1.Kontroler_1_PRM_DIRMP.Done = FALSE) THEN
                                SM1.Execute_1 := TRUE;
                                M1.korak_type := 5;
                        END_IF
                5 : IF SM1.Kontroler_1_PRM_DIRMP.Done THEN
                                M1.korak_type := 10;
                        END_IF
                10: SM1.StartTask := TRUE;
                        SM1.Execute_1 := FALSE;
                        SM1.Write := FALSE;
                        IF SM1.AckStart AND SM1.Kontroler_1_PRM_DIRMP.Done =
FALSE THEN
                                SM1.StartTask := FALSE;
                                M1.korak_type := 20;
                        END_IF
                20: SM1.Execute_1 := TRUE;
                        IF SM1.Kontroler_1_PRM_DIRMP.Done THEN
                                SM1.Execute_1 := FALSE;
                                SM1.Write := TRUE;
                                M1.korak_type := 25;
                        END_IF
                25: IF SM1.Kontroler_1_PRM_DIRMP.Done = FALSE THEN
                                M1.korak_type := 30;
                        END_IF
                30: IF SM1.ActVelocity = SM1.Velocity_1 THEN
                                M1.Promjena_brzine_type := TRUE;
                                M1.korak_type := 40;
                        END_IF
        END_CASE
ELSE M1.korak_type := 0;
END_IF
```

```
(*promjena brzine za kontroler SM2*)
IF M2.Promjena_brzine_type = FALSE THEN
        CASE M2.korak_type OF
                0 : SM2.StartTask := FALSE;
                        SM2.Execute_2 := FALSE;
                        SM2.Write := TRUE;
                        SM2.Velocity_2 := M2.brzina_type;
                        IF (SM2.AckStart = FALSE) AND
(SM2.Kontroler_2_PRM_DIRMP.Done = FALSE) THEN
                                SM2.Execute_2 := TRUE;
                                M2.korak_type := 5;
                        END_IF
                5 : IF SM2.Kontroler_2_PRM_DIRMP.Done THEN
                                M2.korak_type := 10;
                        END_IF
                10 : SM2.StartTask := TRUE;
                         SM2.Execute_2 := FALSE;
                         SM2.Write := FALSE;
                         IF SM2.AckStart AND SM2.Kontroler_2_PRM_DIRMP.Done =
FALSE THEN
                                SM2.StartTask := FALSE;
                                M2.korak_type := 20;
                        END_IF
                20 : SM2.Execute_2 := TRUE;
                        IF SM2.Kontroler_2_PRM_DIRMP.Done THEN
                                SM2.Execute_2 := FALSE;
                                SM2.Write := TRUE;
                                M2.korak_type := 25;
                        END_IF
                25 : IF SM2.Kontroler_2_PRM_DIRMP.Done = FALSE THEN
                                M2.korak_type := 30;
                        END_IF
                30 : IF SM2.ActVelocity = SM2.Velocity_2 THEN
                                M2.Promjena_brzine_type := TRUE;
                                M2.korak_type := 40;
                         END_IF
        END_CASE
ELSE M2.korak_type := 0;
END_IF
(*promjena brzine za kontroler SM3*)
IF M3.Promjena_brzine_type = FALSE THEN
        CASE M3.korak_type OF
                0 : SM3.StartTask := FALSE;
                        SM3.Execute_3 := FALSE;
                        SM3.Write := TRUE;
                        SM3.Velocity_3 := M3.brzina_type;
                        IF (SM3.AckStart = FALSE) AND
(SM3.Kontroler_3_PRM_DIRMP.Done = FALSE) THEN
                                SM3.Execute_3 := TRUE;
```

```
                    M3.korak_type := 5;
                END_IF
            5 : IF SM3.Kontroler_3_PRM_DIRMP.Done THEN
                    M3.korak_type := 10;
                END_IF
            10 : SM3.StartTask := TRUE;
                SM3.Execute_3 := FALSE;
                SM3.Write := FALSE;
                IF SM3.AckStart AND SM3.Kontroler_3_PRM_DIRMP.Done =
FALSE THEN
                    SM3.StartTask := FALSE;
                    M3.korak_type := 20;
                END_IF
            20 : SM3.Execute_3 := TRUE;
                IF SM3.Kontroler_3_PRM_DIRMP.Done THEN
                    SM3.Execute_3 := FALSE;
                    SM3.Write := TRUE;
                    M3.korak_type := 25;
                END_IF
            25 : IF SM3.Kontroler_3_PRM_DIRMP.Done = FALSE THEN
                    M3.korak_type := 30;
                END_IF
            30 : IF SM3.ActVelocity = SM3.Velocity_3 THEN
                    M3.Promjena_brzine_type := TRUE;
                    M3.korak_type := 40;
                END_IF
        END_CASE
ELSE M3.korak_type := 0;
END_IF
```

I.1.8. TIPKE_S();



I.1.9. TIPKA_STOP();

```
IF Pomoćni_stop THEN
        STOP_PRIV();
END_IF
```

I.1.10. STOP_PRIV();

```
SM1.Halt := SM2.Halt := SM3.Halt := FALSE;
PROCEDURA := 50;
UPRAVLJANJE.korakUpr := 0;
SM1.StartTask := SM2.StartTask := SM3.StartTask := FALSE;
```

I.1.11. INIT_S();

```
CASE korak_init OF
        0:      BRZINE_M.M1.brzina_type := SM1.Velocity_1;
                BRZINE_M.M2.brzina_type := SM2.Velocity_2;
                BRZINE_M.M3.brzina_type := SM3.Velocity_3;
                T_init(IN := FALSE);
                T_1(IN := FALSE);
                korak_init := 10;

        10:     T_init(IN := TRUE, PT := T#5S);
                IF T_init.Q THEN
                        SM1.EnableDrive := TRUE;
                        SM2.EnableDrive := TRUE;
                        SM3.EnableDrive := TRUE;
                        SM1.Stop := TRUE;
                        SM2.Stop := TRUE;
                        SM3.Stop := TRUE;
                        T_init(IN := FALSE);
                        korak_init := 20;
                END_IF
        20: IF (SM1.Ready =TRUE) AND (SM2.Ready = TRUE) AND (SM3.Ready =
TRUE) THEN
                        korak_init := 30;
                END_IF
        30: IF (SM1.DriveIsReferenced = FALSE) OR (SM2.DriveIsReferenced = FALSE)
OR (SM3.DriveIsReferenced = FALSE) THEN
                        SM1.StartHoming := TRUE;
                        SM2.StartHoming := TRUE;
                        SM3.StartHoming := TRUE;
                END_IF
```

```
                T_1(IN := TRUE, PT:= t#1s);
                IF T_1.Q THEN
                        SM1.StartHoming := FALSE;

                        SM2.StartHoming := FALSE;
                        SM3.StartHoming := FALSE;
                        korak_init := 20;
                END_IF

                IF (SM1.DriveIsReferenced = TRUE) AND (SM2.DriveIsReferenced =
TRUE) AND (SM3.DriveIsReferenced = TRUE) THEN
                        SM1.StartHoming := FALSE;
                        SM2.StartHoming := FALSE;
                        SM3.StartHoming := FALSE;
                        T_1(IN := FALSE);
                        korak_init := 50;
                END_IF
        50: PROCEDURA := 50;
                korak_init := 0;
END_CASE
```

I.1.12. START_ST();

```
IF SM1.Halt = SM2.Halt = SM3.Halt = FALSE THEN
        SM1.Halt := SM2.Halt := SM3.Halt := TRUE;
         BRZINE_M.M1.Promjena_brzine_type := BRZINE_M.M2.Promjena_brzine_type :=
BRZINE_M.M3.Promjena_brzine_type:= FALSE;
END_IF
```

I.1.13. UPRAVLJANJE();

```
C := DC + Razmak;
IF NovaBrzina_M3 <> BRZINE_M.M3.brzina_type  THEN
        BRZINE_M.M3.brzina_type := NovaBrzina_M3;
        BRZINE_M.M3.Promjena_brzine_type := FALSE;
        NovaBrzina_M2 := 50 * NovaBrzina_M3;
        V1 := (beta*pi*Vor*NovaBrzina_M3) / (C*Wr*180);

        d := (alfa*pi*Vor*NovaBrzina_M3) / (V1*Wr*180);
        d_u_DINT := REAL_TO_DINT(d);
        Nd := Lk - d_u_DINT;
        Ndi := (Nd/55.) * 65536;
        Ndi_DINT := REAL_TO_DINT(Ndi);
        Vcow := REAL_TO_DINT(V1);
        korakUpr := 0;
END_IF
```

```
CASE korakUpr OF
        0 :     IF NovaBrzina_M2 <> BRZINE_M.M2.brzina_type  THEN
                        BRZINE_M.M2.brzina_type := NovaBrzina_M2;
                        BRZINE_M.M2.Promijena_brzine_type := FALSE;
                END_IF
                korakUpr := 10;
        10:
                CRTANJE();
                korakUpr := 20;
        20: NOZ();
                korakUpr := 0;

END_CASE
```

I.1.14. CRTANJE();

```
CASE korakC OF
        0 :     IF SM3.Velocity_3 = SM3.ActVelocity THEN
                        korakC := 10;
                END_IF

        10:     NovaPozicija_M2 := 22010;
                SM2.SetValuePosition := NovaPozicija_M2;
                IF SM2.MC = TRUE THEN
                        SM2.StartTask := TRUE;
                END_IF
                korakC := 20;

        20: PPC := ABS(SM2.ActualPosition - 22010);
                IF PPC <= 5 THEN
                        Detekcija := SM3.ActualPosition;
                        SM2.StartTask := FALSE;
                        b := 1;
                        korakC := 30;
                END_IF

        30:     IF b = 1 THEN
                        D_Crtanje := Detekcija + DCi;
                        Pauza := Detekcija + Cinkr;
                END_IF
                korakC := 40;

        40:     krug := SM3.ActualPosition;
                razlika := ABS(D_Crtanje - krug);
                korakC := 50;
```

```
        50:     IF razlika <= 300 THEN
                        b:= 0;
                        korakC := 52;
                ELSE
                        korakC := 40;
                END_IF

        52: NovaPozicija_M2 := 20005;
                SM2.SetValuePosition := NovaPozicija_M2;
                IF SM2.MC=TRUE THEN
                        SM2.StartTask:=TRUE;
                END_IF
                korakC := 55;
                55: PPP := ABS(SM2.ActualPosition - 20005);
                IF PPP <= 5 THEN
                        korakC:=60;
    END_IF

        60:     krug := SM3.ActualPosition;
                razlika_2 := ABS(Pauza - krug);
                korakC := 70;

        70:     IF razlika_2 <= 300 THEN
                        SM2.StartTask := FALSE;
                        korakC := 80;
                ELSE
                        korakC := 60;
                END_IF
        80:     korakC := 0;
END_CASE
```

I.1.15. NOŽ();

```
krug := SM3.ActualPosition;
CASE korakNoz OF
        0:      IF SM3.Velocity_3 = SM3.ActVelocity THEN
                        korakNoz := 10;
                END_IF
        10:     krug := SM3.ActualPosition;
                IF krug > Ndi_DINT THEN
                        korakNoz := 20;
                ELSE
                        korakNoz := 10;
                END_IF
        20:     NovaBrzina_M1 := Vcow;
                IF NovaBrzina_M1 <> BRZINE_M.M1.brzina_type THEN
                        BRZINE_M.M1.brzina_type := NovaBrzina_M1;
```

```
                BRZINE_M.M1.Promjena_brzine_type := FALSE;
                korakNoz := 40;
            END_IF

END_CASE
```

I.2. CoDeSys v3.5

```
xUpaljeno := TRUE;

fbDelay_paljenja_plc(IN:= xUpaljeno, PT := T#10MS);

IF fbDelay_paljenja_plc.Q THEN

    LGV.xSM_1_Noz_Din4 := TRUE;
    LGV.xSM_2_Olovka_Din4 := TRUE;
    LGV.xSM_3_Namatalica_Din4 := TRUE;
    LGV.xSvi_SM_DIN5 := TRUE;
END_IF
```

II. TEHNIČKA DOKUMENTACIJA